Pedro Filipe Maravilha Moreira

Licenciado em Engenharia Eletrotécnica e de Computadores

Gestor de políticas de autorização e de acesso numa rede empresarial

Dissertação para obtenção do Grau de Mestre em Engenharia Eletrotécnica e de Computadores

Orientador: Paulo da Costa Luís da Fonseca Pinto, Professor Catedrático, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Março, 2017

Gestor de políticas de autorização e de acesso numa rede empresarial

Agradecimentos

Gostaria de agradecer a todas as pessoas que me ajudaram durante o desenvolvimento desta dissertação.

Um obrigado ao Prof. Paulo Pinto por toda a ajuda e pela sua disponibilidade para orientar este projeto.

Agradeço também à minha namorada e aos colegas que me auxiliaram e acompanharam ao longo deste projeto e do curso.

Um agradecimento também para a minha família, especialmente para os meus pais e para a minha irmã, sobrinho e cunhado.

Resumo

O estudo realizado no âmbito desta tese de Mestrado centra-se no desenvolvimento de uma solução de controlo de acesso ao nível empresarial. A solução em questão baseia-se e foi desenvolvida no contexto do *Software Defined Networking*, uma área emergente no que diz respeito a redes de computadores com especial ênfase na configuração e na gestão. O estudo realizado engloba as diferentes fases de desenvolvimento de uma solução de controlo de acesso baseada em *Software Defined Networking* e baseia-se na criação de um modelo de controlo de acesso que pode ser facilmente adaptado a diferentes tipos de redes e de aplicações.

O modelo de controlo de acesso desenvolvido tem por base a criação de perfis e de políticas de controlo de acesso. Trazendo para esta solução o conceito de *Role-based Access Control*, ele possibilitou que os acessos permitidos e não permitidos numa rede, para um determinado utilizador, sejam definidos com base no *Role* atribuído a esse mesmo utilizador. O âmbito das políticas e perfis apresentados não se restringe ao controlo de acesso clássico, englobando também outras áreas relevantes como a da qualidade de serviço.

A última fase do processo engloba a implementação prática do modelo, os seus perfis e as suas políticas e o teste do funcionamento deste modelo. A implementação e os testes foram realizados com recurso a *software* e *hardware* desenvolvidos e comercializados pela HP.

Palavras-chave: *Software Defined Networking*, Controlo de Acesso, OpenFlow, Autenticação, Redes Empresariais

Abstract

The study carried out during the course of this Master's thesis focuses on the development of an access control solution to be deployed at the enterprise level. This solution is developed using Software Defined Networking technology. Software Defined Networking is an emerging area related to computer networks with special emphasis in network configuration and network management. This study encompasses the different stages of development of a Software Defined Networking access control solution and creates an access control model that can be easily adapted to different types of networks and applications.

The access control model is defines the creation of profiles and access control policies. By adopting the concept of Role-based Access Control, the access allowed in a network, for a given user, is defined according to the profile assigned to him/her. The scope of the presented policies and profiles is not restricted to access control, but also encompasses other relevant areas such as quality of service.

The practical implementation of the model is the last stage of the process, defining its profiles and policies, and the test of how this model works. Implementation and testing were performed using software and hardware developed and marketed by the Hewlett-Packard company.

Keywords: Software Defined Networking, Access control, OpenFlow, Authentication, Enterprise Networks

Índice

Índice de Figuras

Índice de Tabelas

Siglas

API Application Programming Interfaces

DoS Denial-of-service

EAP Extensible Authentication Protocol

FTP File Transfer Protocol

HTTP Hypertext Transfer Protocol

IEEE Institute of Electrical and Electronics Engineers

LAN Local Area Network

LLDP Link Layer Discovery Protocol

MAC Media access control

PKI Public key infrastructure

QoS Quality of Service

RADIUS Remote Authentication Dial In User Service

R&D Research and Development

SDN Software Defined Networking

SMTP Simple Mail Transfer Protocol

SSH Secure Shell

SSL Secure Sockets Layer

TCP Transmission Control Protocol

TLS Transport Layer Security

VAN Virtual Application Networks

VLAN Virtual LAN

VoIP Voice over IP

XML eXtensible Markup Language

1 Introdução

O conceito de *Software Defined Networking* (SDN) permite que as redes sejam controladas a partir do seu exterior permitindo que tenham um comportamento funcional ditado por aplicações. As aplicações trabalham com conceitos de alto nível que têm de ser traduzidos para outros de mais baixo nível, compatíveis com os controladores SDN. Estes controladores, por sua vez, gerem um conjunto de equipamentos de rede (*switches* por exemplo) pondo em prática as orientações das aplicações.

Como se irá ver, o SDN está ainda muito focado em poucos conceitos, que embora já relevantes, não permitem uma gama muito grande de intervenção na rede. Os conceitos mais importantes são o estabelecimento de fluxos, e o manuseamento de fluxos no sentido de se lhes poderem alterar algumas características. Outros aspetos que são cobertos, mas não de um modo muito eficiente prendem-se com monitorização do tráfego que passa e prioridade básica de serviço de pacotes. É possível executar algumas tarefas desde que se possa trabalhar com valores médios. Finalmente, aspetos relacionados com tempo real, como larguras de banda, latências, mecanismos de serviço de filas de espera e suas dimensões ainda não estão cobertos de um modo aceitável nas versões atuais de SDN.

O objetivo deste projeto é desenvolver uma solução de autenticação e controlo de acesso em redes de telecomunicações, no contexto do SDN. Pretende-se analisar a exequibilidade de tal solução no estado atual de definição do SDN.

A solução desenvolvida é um protótipo de uma unidade de gestão de acessos a endereços e serviços numa rede empresarial. A unidade de gestão reside no controlador de SDN e é configurada a partir de uma aplicação exterior. Ela recebe informação de identificação do utilizador na altura em que este se regista na rede e interage com os *switches* SDN da rede com o propósito de monitorizar a atividade do utilizador de acordo com perfis definidos e de autorizar ou rejeitar interações com outros elementos da rede. Para atingir estes objetivos o projeto desenvolve-se segundo as seguintes três linhas que permitem implementar um bom mecanismo de autenticação e gestão de acessos para redes SDN.

Em primeiro lugar, é criado um modelo teórico de controlo de acesso, através da definição de um conjunto de políticas e de perfis. O modelo define, numa rede, os fatores necessários para que um utilizador aceda a recursos específicos da mesma especificando quem pode aceder a que recursos da rede. Em seguida, a formalização do modelo é realizada. O modelo que definimos tem por base um modelo já existente, que é detalhado no segundo capítulo.

Em segundo lugar o modelo que formalizámos teoricamente é implementado utilizando o controlador HP VAN SDN para definir políticas ao nível do fluxo (*flow*). A definição dos campos e dos valores a comparar (os *matches*) de cada fluxo vai permitir, por exemplo, rejeitar o tráfego de um utilizador que não se tenha autenticado corretamente, ou tratar o tráfego de cada utilizador de forma diferente consoante o tipo de perfil que este possua.

A estrutura da tese é a seguinte: no segundo capítulo é apresentado algum do trabalho relacionado havendo uma especial atenção com a gestão de controlo de acesso no domínio SDN. O terceiro capítulo descreve a implementação do modelo definido, estruturada com base nas várias etapas de desenvolvimento, desde a definição teórica da solução, à formalização da mesma com base num modelo existente e finalizando com a implementação com recurso ao controlador HP VAN SDN. No quarto capítulo apresentam-se os resultados obtidos através de testes de simulações

realizando-se um sumário das conclusões a que se chegou após o desenvolvimento e teste da solução, assim como direções para trabalho futuro, no capítulo final.

2 Estado da Arte

2.1 Software Defined Networking

2.1.1 Arquitetura SDN

Em [1] o *Software Defined Networking* é definido como uma arquitetura de rede em que o plano de dados e o plano de controlo estão separados. Esta separação introduz uma nova forma de controlar, modificar e gerir os comportamentos de uma rede. Em vez de serem utilizadas interfaces criadas pelos fabricantes, que construíram os dispositivos que constituem a rede, o controlo dos comportamentos da rede é realizado através de *standards* públicos, definidos como *open interfaces*.

Uma das maiores vantagens do *Software Defined Networking* é a sua transversalidade relativamente aos dispositivos que podem ser utilizados. Podemos introduzir um controlo centralizado numa rede assente numa infraestrutura composta por dispositivos de fabricantes diferentes. Os próprios equipamentos da rede também deixam de ter que lidar com um número enorme de protocolos distintos.

As aplicações SDN (*Software Defined Networking)* podem utilizar a *northbound* API, suportada pelo plano de controlo, para fazer cumprir as suas políticas no plano de dados sem interagirem diretamente com o mesmo [1]. A interface entre o plano de controlo e o plano de dados é suportada pelas *southbound* APIs e o controlador utiliza estas APIs para comunicar com os equipamentos da rede.

O SDN permite também o *virtualized networking* e serviços de *cloud* seguros.

Em [2] surgem, aquilo que podem ser caracterizados como os quatro pilares de uma arquitetura *Software Defined Networking* (arquitetura essa representada na Figura 1):

1. O desacoplamento entre os planos de dados e de controlo e a remoção da funcionalidade de controlo dos dispositivos da rede.
2. As decisões de encaminhamento são baseadas em *flows*. No contexto SDN, uma *flow* é uma sequência de pacotes transmitidos entre uma origem e um destino.
3. A lógica de controlo é movida para uma entidade externa, o controlador SDN. O controlador SDN é uma plataforma de *software* que fornece os recursos e abstrações essenciais para a programação dos dispositivos de encaminhamento.
4. A rede é programável, através de aplicações de *software* que correm sobre o controlador.

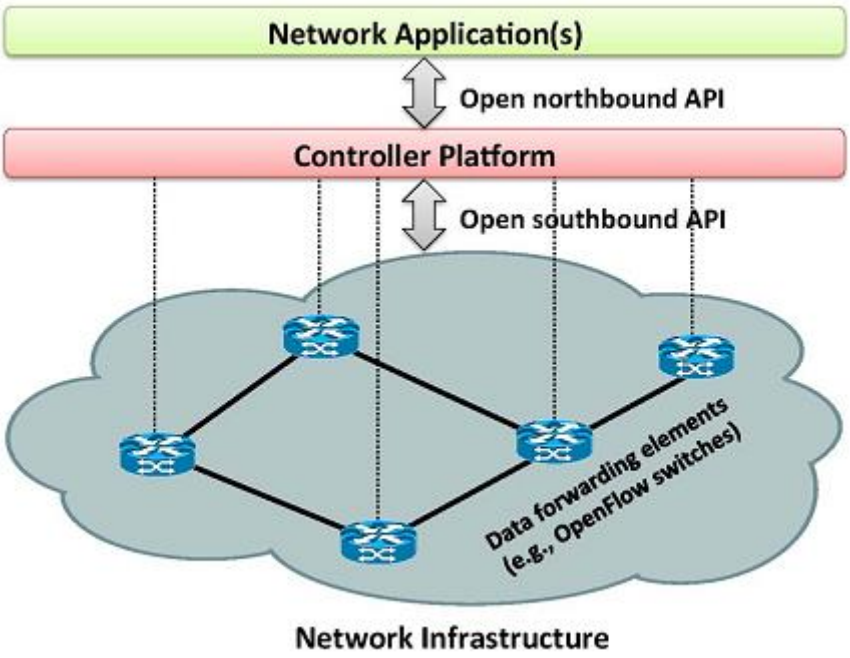

Figura 1: Arquitetura SDN [2]

O conceito *Software Defined Networking* pode também ser definido por três abstrações fundamentais [2]: encaminhamento, distribuição e especificação. A abstração de encaminhamento deve esconder os detalhes de *hardware*, enquanto permite qualquer comportamento de encaminhamento desejado pela aplicação de rede. A abstração de distribuição deve tornar o problema do controlo distribuído num problema lógico centralizado. A implementação desta abstração requer uma camada comum de distribuição com duas funções. Instalar os comandos de controlo nos dispositivos de encaminhamento e recolher dados sobre os dispositivos de rede e as ligações. A abstração de especificação deve permitir à aplicação de rede definir um determinado comportamento para a rede sem ser responsável realmente por implementar esse comportamento. Esta abstração é alcançada através das linguagens de programação.

A arquitetura SDN é composta por vários níveis (*layers*) ou camadas [2], cada uma com as suas funções específicas. No entanto, alguns desses *layers* não estão presentes em todas as situações que possamos encontrar, como é o caso do *hypervisor* ou da virtualização baseada em linguagem.

A denominada *Layer I*: Infraestrutura [2] é composta pelo equipamento utilizado na rede, como acontece para as redes tradicionais, como *switches* e *routers*. A diferenciação principal, relativamente às redes tradicionais é a simplificação destes dispositivos. Ou seja, eles podem atuar como simples encaminhadores, trabalhando apenas no plano de dados e o "cérebro" da rede é um controlador, como o controlador HP VAN SDN utilizado neste projeto. Estes simples encaminhadores serão programados dinamicamente, não esquecendo que são dispositivos heterogéneos. Podem ser encaminhadores que utilizem o *standard* OpenFlow que é referido na próxima secção. A segunda camada, *Layer II* são as Interfaces *Southbound* e dizem respeito às APIs que ligam os elementos de controlo aos elementos de encaminhamento, separando o plano de controlo do plano de dados. Os *standards* introduzidos em termos de interfaces *Southbound* promovem a interoperabilidade permitindo a utilização de dispositivos de fabricantes diferentes. Os *Network Hypervisors* constituem a *Layer III* e são os *Hypervisors* que permitem que máquinas virtuais distintas partilhem os mesmos recursos de *hardware*, o que faz com que cada utilizador

possa ter os seus recursos virtuais, em termos de computação ou de armazenamento. A *Layer IV* inclui os controladores da rede. Esta camada define alguns dos elementos mais importantes da rede.

O controlador é mesmo um dos elementos críticos na arquitetura SDN, visto que é o elemento que suporta o controlo lógico para gerar a configuração baseada nas políticas definidas. A plataforma de controlo, neste caso o controlador, assemelha-se a um sistema operativo tradicional, no sentido em que esconde os detalhes de mais baixo nível relacionados com as ligações com os dispositivos de encaminhamento.

O controlo pode ser centralizado ou distribuído. Um controlo centralizado caracteriza-se por um único controlador, que gere todos os dispositivos de encaminhamento da rede. Se a rede for escalando este tipo de controlo apresenta as suas limitações. Para além disso, o controlador é sempre um ponto único de falha tornando a solução pouco resiliente. O controlo centralizado expõe a rede a um risco elevado pois é o alvo de qualquer ataque que tenha por objetivo a manipulação dos serviços comuns da rede, ou até o controlo total da rede a partir do controlador [3]. Com o aumento do número de dispositivos de encaminhamento do plano de dados podem ser necessários mais controladores. O controlo centralizado é normalmente utilizado em ambientes como *data centers* e infraestruturas de *cloud* [2].

Um controlo distribuído, pelo contrário, consegue ser escalado para qualquer ambiente, desde redes de pequena escala a redes de larga escala. O controlo distribuído pode ser feito de diferentes formas, através de um conjunto centralizado de controladores, ou através de um conjunto de controladores distribuídos fisicamente pela rede. Sendo esta segunda opção mais resistente a falhas [2].

2.1.2 OpenFlow

O protocolo OpenFlow é o que normalmente se usa na interface *Southbound*. É o protocolo utilizado para a comunicação entre o controlador e um *switch* compatível. Esta comunicação pode usar um canal seguro para uma melhor proteção. Os *switches* compatíveis com OpenFlow apresentam um conceito de várias tabelas de *flows* que podem ser configuradas a partir das primitivas/mensagens definidas no protocolo. Existem ainda outros conceitos como portas lógicas, grupos de *flows*, entre outros. A mensagem mais importante é a definição de uma entrada na tabela. Ela contém os campos a serem testados para verificar se são iguais aos do pacote e a ação a ser executada. Toda essa informação constitui uma entrada da tabela. Em funcionamento normal, aquando da receção de um pacote, o *switch* OpenFlow pode tomar uma de três decisões: pode fazer o encaminhamento do pacote para um porto; pode encaminhar o pacote para o controlador através da ligação segura; ou pode descartar o pacote. A escolha da ação respetiva está armazenada na tabela. Encaminhar muitos pacotes para o controlador leva a um atraso muito grande na rede. Faz sentido no caso de ser o primeiro pacote de uma *flow*. O controlador fica a saber os vários parâmetros e pode depois instalar uma nova entrada na tabela. O modelo OpenFlow está representado na Figura 2.

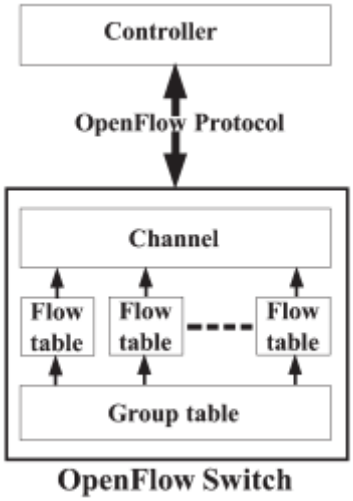

Figura 2: Modelo OpenFlow [4]

Cada entrada de uma tabela de *flows* possui três partes: uma regra de *matching*, as ações a serem executadas para os pacotes que façam o *matching* e contadores que mantêm um registo dos pacotes que fizeram *matching*. Este modelo é o mais difundido para dispositivos do plano de dados.

Um dispositivo OpenFlow pode ter mais do que uma tabela. As tabelas dispõem-se em níveis. As passagens de um nível para outro são feitas a partir de ações nas entradas da tabela. É possível pensar-se num caminho através de uma sequência de tabelas de *flows* que determina o que deve ser feito com os pacotes: mais uma vez, se devem ser encaminhados na rede, se devem ser descartados ou então enviados para o controlador. O processo de análise do pacote começa na primeira tabela de *flows* e termina quando é, ou não, encontrado um *match*. Uma regra presente numa tabela pode ser definida por vários campos de *matching*, como é ilustrado na Figura 3. Pode não ser encontrada nenhuma regra para o pacote em questão. Para contemplar estes casos, é normalmente instalada uma regra *default*, para que quando o pacote não fizer *matching* com nenhuma regra, seja enviado para o controlador (ou descartado).

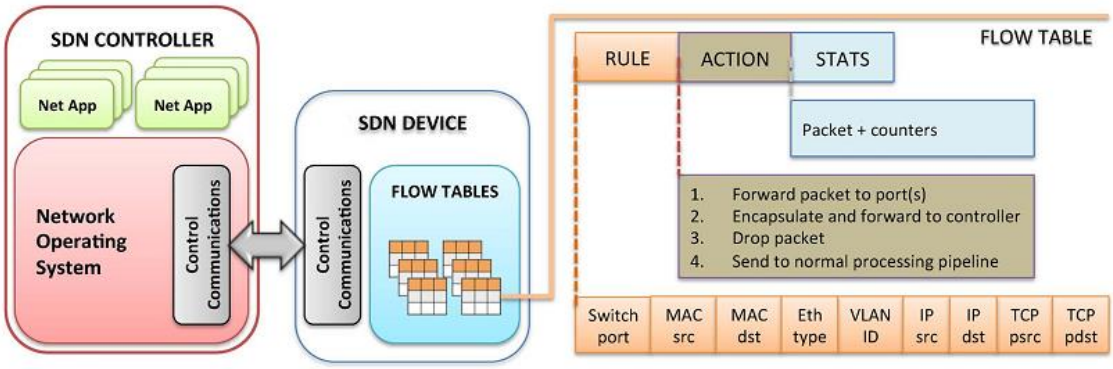

Figura 3: Dispositivo OpenFlow [2]

O protocolo OpenFlow fornece três fontes de informação para os controladores [2]: mensagens baseadas em eventos enviadas dos *switches* para o controlador; estatísticas de *flows* geradas pelos encaminhadores e guardadas pelo controlador; e mensagens *packet-in* enviadas para o controlador

quando o encaminhador não sabe o que fazer com um pacote, ou quando a ação é especificamente para o enviar para o controlador. Apesar de o OpenFlow ter grande aceitação, existem alternativas como: Open vSwitch Database [5], POF [6], OpFlex [7], entre outros.

Como foi referido, a comunicação entre o controlador e o encaminhador é estabelecida através de um canal seguro, designado por canal OpenFlow. É o canal por onde o controlador configura e gere o *switch*. O protocolo suporta três tipos de mensagens [8]: *controller-to-switch*, assíncronas e simétricas. Cada um destes tipos tem subtipos definidos.

As mensagens *controller-to-switch* são iniciadas pelo controlador com o objetivo de obter a informação sobre o estado do *switch*. Para fazer uma atualização em relação aos eventos que ocorrem na rede e às mudanças de estado do *switch*, o *switch* pode iniciar a transmissão de mensagens assíncronas. As mensagens síncronas tanto podem ser iniciadas pelo controlador, como pelo *switch*. O canal OpenFlow normalmente está encriptado com a tecnologia TLS, mas também pode funcionar diretamente sobre TCP.

2.1.3 Segurança

Ao desenvolver uma solução relacionada com a autenticação e o controlo de acesso em SDN, deve-se ter em conta os problemas de segurança existentes. Em termos de segurança em SDN, existem vários desafios com que nos deparamos, alguns dos mais importantes são os possíveis ataques de negação de serviço e as vulnerabilidades que cada componente da rede pode apresentar.

Neste tipo de redes podem se encontrar alguns vetores [9] que representam as ameaças possíveis. Uma dessas ameaças é a falsificação de *flows*, que pode ter como objetivo atacar *switches* e controladores. Um atacante pode lançar um ataque *Denial-of-Service* (DoS) com *flows* falsas. Quando um atacante assume o controlo de um servidor pode retirar informação sobre os utilizadores e depois injetar na rede *flows* maliciosas. Utilizando um porto autenticado e um endereço MAC de origem que seja aceite, o ataque torna-se mais fácil.

O ataque DoS também pode ser feito quando um atacante ganha acesso ao plano de controlo e pode controlar *switches* suficientes para consumar esse ataque. Para além disso o ataque pode ser feito tendo em conta as vulnerabilidades dos *switches*, sendo que um único *switch*, pode ser usado para desviar tráfego ou injetar pedidos forjados para sobrecarregar o controlador e os *switches* vizinhos. Para solucionar esse problema poderia ser implementado um mecanismo para monitorizar e detetar comportamentos anormais dos dispositivos da rede [9].

Os ataques diretos ao controlador apresentam uma grande ameaça, visto que, podem permitir ao atacante controlar a rede. Nestes casos o problema reside na dificuldade em detetar a exata combinação de eventos que dá origem a um comportamento em particular. Uma aplicação maliciosa pode potencialmente fazer tudo o que quiser já que os controladores apenas disponibilizam abstrações que se traduzem em comandos de configurações para os *switches* da rede. Em redes deste tipo deve-se assegurar a integridade dos elementos mais sensíveis dentro do controlador, como por exemplo, as chaves criptográficas [9].

Outra questão que se coloca é como assegurar a confiança entre o controlador e as aplicações de gestão. Estas duas entidades não têm a capacidade de estabelecer relações confiáveis porque as técnicas utilizadas para certificar aplicações são diferentes das usadas para certificar dispositivos de rede.

Para criar uma rede SDN segura é importante também que as deficiências e as fraquezas presentes nas arquiteturas das redes tradicionais, que possam ser exploradas, não sejam repetidas nesta rede SDN. Do ponto de vista de programação da rede também podem surgir novas ameaças, visto que o SDN oferece um acesso de programação explícito aos clientes que são tipicamente entidades organizacionais separadas. Esta alteração apresenta requisitos de segurança que não existiam nas redes tradicionais, dentro de ambientes administrativos fechados, em termos da integridade do sistema e de *open interfaces* [3].

Em termos de técnicas desenvolvidas para tentar assegurar redes seguras, uma das mais importantes é a replicação. Fazendo a replicação do controlador e também das próprias aplicações, tendo todas as aplicações em cada réplica do controlador. Esta abordagem, permite também combater melhor as falhas de *hardware* e *software*. Outra técnica relevante para melhorar a robustez de redes seguras é a diversidade. Diversidade neste contexto significa utilizar diferentes controladores e não apenas réplicas de um controlador específico. A utilização de controladores diferentes, permite evitar algumas vulnerabilidades e *bugs* de *software*. A introdução de diversidade na rede só é possível porque em *SDN* podemos ter a mesma aplicação de gestão e controlo a correr em controladores diferentes. Podem ser também usados mecanismos que possibilitem uma recuperação proactiva, substituindo componentes que tenham sido comprometidos do ponto de vista de segurança [9].

A associação dinâmica de dispositivos também é uma técnica preponderante. Um *switch* deve conseguir associar-se dinamicamente a vários controladores de uma forma segura. Um *switch* associado a vários controladores estaria apto a tolerar falhas automaticamente. A deteção de controladores maliciosos também deve ser feita e a autenticação deve impedir a existência de um ataque *man-in-the middle*. A autenticação é determinante para assegurar a confiança entre os dispositivos e os controladores.

2.2 Sistemas e *frameworks* de autenticação e controlo de acesso existentes

Relativamente a mecanismos de autenticação, ou sistemas de controlo de acesso em redes, existem já várias propostas, com diferentes abordagens. Alguns sistemas de autenticação e controlo de acesso foram desenhados para assegurar a segurança de redes empresariais, como por exemplo o Resonance [10]. Resonance é um *framework* que fornece mecanismos para implementar dinamicamente políticas de segurança de rede nos dispositivos. Este é um conceito útil principalmente porque deixa pouca responsabilidade para os *hosts* e para as *layers* superiores da rede. A inspiração para esta ideia surgiu do desenho de sistemas operativos seguros, visto que, num sistema operativo encontramos uma construção de sistemas complexos utilizando pequenos componentes como base.

Este *framework* permite dotar a rede de funções básicas, que são necessárias para a implementação de políticas de segurança. Tal como em muitas outras propostas (algumas delas apresentadas mais adiante) o tráfego é controlado utilizando políticas que um controlador instala nos *switches* OpenFlow. Este tipo de filosofia permite definir um *framework* constituído pelo controlador e pelos restantes dispositivos. Se as condições na rede da empresa em questão mudarem, como é natural que aconteça, facilmente se pode modificar a forma como a rede deve controlar o tráfego.

O Resonance apresenta também um recurso interessante, que possibilita proteger alguns *hosts*, quando tiver sido detetada uma falha de segurança na rede. Um conjunto de *hosts* importante pode ser "isolado" dos restantes. Este procedimento envolve a especificação de uma política de

segurança e é preciso definir que tipo de tráfego pode indicar a existência de uma falha de segurança num determinado *host*. A monitorização da rede é, no Resonance, distribuída e intrínseca à arquitetura, mas existe um mecanismo que permite uma visão geral da rede, visto que os dispositivos da rede podem encaminhar informação sobre o tráfego da rede para uma localização centralizada para inferir o estado da rede. Deste modo, o Resonance monitoriza a rede, não de uma forma individual, que seria através da análise do comportamento de cada *host* e dos padrões do seu tráfego, mas sim de uma forma coordenada. Os seus autores reclamam que este tipo de aproximação é também mais adaptado a identificação de ataques coordenados.

O objetivo desta proposta e de outras com o mesmo propósito é o de possibilitar um controlo mais fino através das políticas de rede e tentar explorar outras funcionalidades que possam ser incorporadas do ponto de vista da autenticação e do controlo de acesso à rede. Neste sistema são os *switches* que fazem cumprir as políticas de segurança. Ou seja, o Resonance permite implementar diretamente políticas de segurança de rede nos *switches*. No Resonance, o próprio nível rede apresenta funções básicas para implementar as políticas de segurança. No entanto, também é implementada uma interface de controlo que permite fazer o controlo de tráfego de acordo com as políticas pré-definidas. As funcionalidades mais relevantes da arquitetura Resonance são: o *framework* de especificação de políticas, a monitorização distribuída da rede e a capacidade de tomar ações usando os *switches* OpenFlow [10].

Para além do Resonance, existem propostas como o AuthFlow [11] que tem como característica mais importante o facto de autenticar *hosts* no nível *data link* numa rede OpenFlow. Isto introduz uma baixa sobrecarga (*overhead*) e assegura um controlo de acesso fino. O atraso é menor do que o que seria introduzido se a autenticação fosse feita ao nível rede ou ao nível aplicação.

Em [12] é definido o mecanismo Ethane, que controla a rede ao requerer a existência de uma permissão especifica para que dois *hosts* possam comunicar. A política global da rede está presente no controlador, que decide se a *flow* representada pelo pacote em questão deve ser permitida. Para as *flow*s permitidas, o controlador calcula o melhor caminho, dado que conhece a topologia da rede. Para construir a topologia da rede o controlador agrega a informação que recebe dos *switches*, que se autenticam perante o controlador e estabelecem uma ligação segura à medida que a *spanning tree* é criada.

Os *switches* do Ethane são simples encaminhadores, com uma tabela de *flows* e um canal seguro para o controlador. A linguagem de políticas Pol-Eth foi desenvolvida [12] em conjunto com o mecanismo Ethane, Um conjunto de políticas foram declaradas assim como um conjunto de regras. Estas regras consistem em predicados e em ações que ocorrem para as *flow*s correspondentes.

Com o Ethane, todos os *switches*, utilizadores e *hosts* estão registados no controlador com as credenciais necessárias para os autenticar. Os *switches* são pré-configurados com as credenciais necessárias para autenticar o controlador, como por exemplo, a chave pública do controlador.

2.2.1 Autenticação

Como se viu, em SDN o controlo da rede é centralizado e independente da tecnologia utilizada para conectar os dispositivos da rede. A autenticação numa rede SDN pode ser realizada implementando o mecanismo de funcionamento do *standard* para controlo de acesso às redes tradicionais, o protocolo 802.1X.

Para implementar o funcionamento do protocolo 802.1X, é utilizado um *switch* autenticador e um servidor RADIUS. O servidor RADIUS transmite ao autenticador a informação que possibilita que este permita, ou não, ao utilizador o acesso à rede. Numa primeira fase o utilizador (tipicamente designado de *Supplicant*) envia um pedido ao autenticador, que recorre ao servidor RADIUS para fazer o processo de autenticação. Depois, com base na informação recebida do servidor RADIUS, o autenticador permite que o utilizador aceda, ou não, à rede. Colocando o controlador e os *switches* entre o *Supplicant* e o servidor RADIUS, a lógica de autenticação pode ser definida no controlador, o que permite uma maior flexibilidade. Quando a lógica de autenticação é definida apenas num *switch* autenticador, torna-se mais difícil alterar as políticas de acesso ao meio por não haver flexibilidade.

Exemplificando o processo, o AuthFlow [11] faz a autenticação no nível data *link* com o protocolo 802.1X, o que faz com que não seja necessário efetuar qualquer alteração nos *hosts*. A informação trocada é encapsulada em EAP (*Extensible Authentication Protocol*), o que permite a utilização de vários métodos de autenticação diferentes. A arquitetura do AuthFlow consiste num controlador OpenFlow e dois outros componentes, o autenticador e o servidor RADIUS. O autenticador recebe mensagens 802.1X e valida as credenciais perante o servidor RADIUS. O controlador tem que correr uma aplicação para tratar o encaminhamento de pacotes, em particular os pacotes 802.1X. Estes pacotes são enviados diretamente para o autenticador. O autenticador é um cliente RADIUS que implementa o 802.1X e envia mensagens EAP para o servidor.

O cliente RADIUS envia para o controlador uma confirmação de sucesso de autenticação através de um canal encriptado usando SSL 3.0 e PKI (*Public key infrastructure*). O Servidor RADIUS extrai a informação encapsulada em EAP e valida as credenciais apresentadas com uma base de dados. Se as credenciais estiverem corretas, o autenticador envia uma mensagem de sucesso para o *Supplicant host* e envia uma autorização e mensagem de confirmação para o controlador, através de um canal seguro SSL. Esta mensagem identifica o *Supplicant* pelo seu endereço MAC e confirma o sucesso da autenticação.

Após a autenticação, o controlador permite ao *host* aceder, ou não, aos recursos da rede. As credenciais dependem dos mecanismos de autenticação utilizados. Os *hosts*, por exemplo, podem ser autenticados pelos seus endereços MAC, os utilizadores pelo nome de utilizador e por uma palavra-passe e os *switches* através de certificados de segurança.

Na Figura 4 é ilustrada a arquitetura AuthFlow, com os seus três componentes principais: o controlador OpenFlow, o autenticador e o servidor RADIUS. Está também ilustrada a máquina virtual que começa a autenticação de acordo com o standard IEE 802.1X e as mensagens de autenticação que vêm nos pacotes EAP para a realização da autenticação.

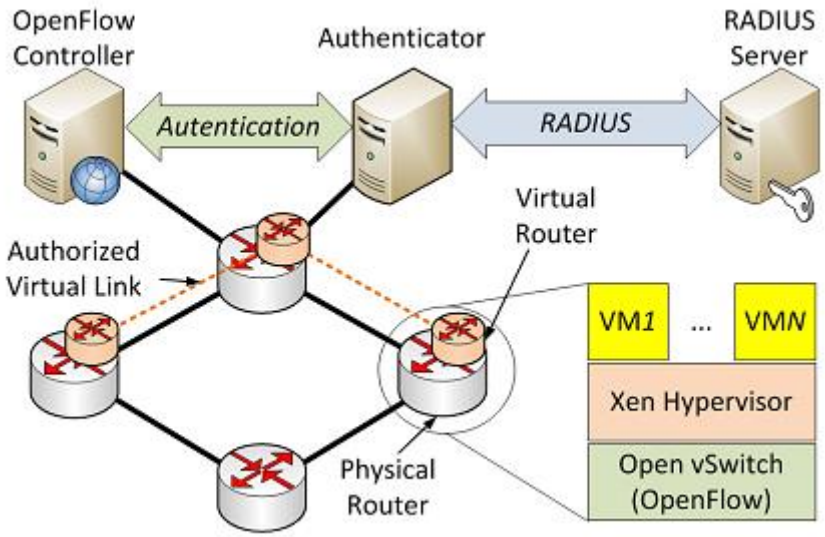

Figura 4: Arquitetura AuthFlow [11]

2.2.2 Controlo de Acesso

Com recurso a um controlador e a *switches* OpenFlow, o controlo de acesso é mais eficiente do que nas redes tradicionais. A capacidade existente em soluções como o Resonance [10] de especificar e implementar um controlo de acesso dinâmico é totalmente diferente das que encontramos em configurações de redes tradicionais. Numa rede tradicional sem a introdução de um controlador e *switches* OpenFlow, os utilizadores pertencentes a um grupo de segurança específico, estariam todos atribuídos à mesma VLAN. Logo todo o tráfego numa VLAN estava sujeito a uma política e os *hosts* numa VLAN não estariam protegidos dos restantes *hosts* dessa VLAN. Para além disso, a alteração da distribuição de utilizadores por VLAN tem que ser alterada manualmente, ao contrário do que se pode fazer num cenário SDN em que o controlo de acesso pode ser feito dinamicamente.

O AuthFlow [11] permite definir políticas de acesso baseadas em *flows* para cada *host* de acordo com uma credencial. O controlo de acesso pode ser feito de acordo com o nível de privilégios de cada *host*, emparelhando as credenciais do *host* e um conjunto de *flows* que pertencem ao *host*. A definição dos serviços a que cada utilizador pode aceder é feita a partir das credenciais e não a partir dos endereços IP ou MAC. Na prática, o controlo de acesso fornece ou nega o acesso das máquinas virtuais aos *links* virtuais. Logo, quando se inicia o funcionamento de uma rede OpenFlow com AuthFlow, todas as ligações da rede estão inicialmente bloqueadas para qualquer tipo de comunicação, incluindo as ligações do *core* da rede, que interligam *switches* OpenFlow. A eficiência deste tipo de controlo de acesso é confirmada em experiências apresentadas pelos autores do AuthFlow [11].

Note-se que para realizar o controlo de acesso poderíamos apenas analisar a *flow* em questão e verificar o campo referente ao endereço MAC de quem a originou, juntamente com o *incoming port*. No entanto, verificando se estes dois campos da *flow* correspondem aos dados de autenticação, as credenciais podem ser atribuídas à *flow*, correlacionado assim a identidade de um *host* com a *flow*. Assim temos um controlo mais fino, feito a partir da credencial do *host* e não apenas a partir de dois campos da *flow*.

2.2.3 Perfis e estados

No nosso modelo são definidos perfis de autenticação. Existem já propostas que incorporam conceitos similares, como no Resonance [10] em que cada *host* tem uma espécie de perfil atribuído, denominado de classe. Para além disso, cada *host* tem também um estado atribuído. O estado pode modificar-se ao longo do tempo de acordo com um conjunto de transições, que são, por sua vez, definidas por políticas. Ou seja, o estado de um *host* vai se modificando, sendo as transições de estado definidas pelas políticas. As políticas no fundo definem o que fazer com o tráfego de um determinado *host*, com uma determinada classe e estado. A classe é atribuída a um determinado *host* consoante os recursos da rede que lhe queremos disponibilizar. Uma política determina as ações que um *switch* toma com o tráfego. O facto de as políticas serem dinâmicas possibilita que os *switches* modifiquem a forma como controlam o tráfego de um *host* à medida que as condições da rede se alteram.

No Resonance as classes permitem-nos estabelecer o tipo de acesso que um utilizador tem aos recursos da rede. As classes formalizam também até que ponto um *host* na rede é monitorizado.

Na rede do campus de uma universidade, os utilizadores que se autentiquem na rede como *Guests*, podem ter por exemplo a limitação de apenas poderem transmitir tráfego para a internet. Isto protege os estudantes da universidade que estão a utilizar a rede e pode ser implementado com recurso a classes ou perfis.

O *framework* [10] especifica os estados possíveis para cada *host* (representados na Figura 5). Estão também representadas as políticas de controlo de acesso, as ações que os dispositivos da rede devem tomar e a definição de como os *hosts* podem transitar de um estado para outro. Cada classe apresenta um conjunto pré-definido de estados e um conjunto de transições possíveis. Os *switches* podem utilizar múltiplas tabelas para um *host*, mas a tabela a utilizar depende do estado atual e da classe do *host*.

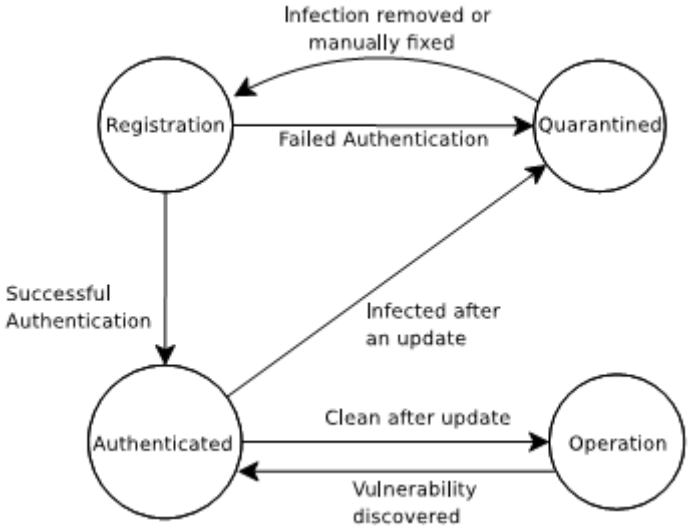

Figura 5: Transições de estado no Resonance [10]

Analogamente ao Resonance, o FortNox [13] não implementa classes, mas implementa *roles*. Para que seja feita uma autenticação da fonte do tráfego o FortNox define três *roles* para atribuir

aos agentes que produzem os pedidos de inserção de uma regra [13]. Uma regra OpenFlow corresponde no fundo, a uma entrada na tabela de fluxos que define *matches*, *instructions* e *counters*.

Os pedidos de inserção de regra com maior prioridade são os originários de utilizadores a quem foi atribuído o *role* de *human administrators*. O mecanismo de resolução de conflitos do FortNox interpreta estes pedidos com uma prioridade superior à dos pedidos provenientes de outros *roles*. As aplicações de segurança também possuem um *role* próprio para que possam lidar com ameaças em tempo real, que não podem ser abordadas da mesma forma por um administrador da rede. Por exemplo, uma *flow* maliciosa deve ser detetada pelas aplicações de segurança. No entanto a prioridade dos pedidos provenientes dos administradores é ainda superior à prioridade dos pedidos que originam das aplicações de segurança. A todas as outras aplicações, que não estão relacionadas com a segurança da rede é associada a prioridade mais baixa para os seus pedidos.

No Authflow [11], são utilizados estados para estabelecer todo o processo de autenticação e controlo de acesso. Quando um *host* não está autenticado, todo o trafego gerado ou enviado para este *host* é ignorado exceto o tráfego *ethernet* 0x888E (IEEE 802.1X). O tráfego 802.1X é enviado do *host* para o autenticador como uma *flow multicast* e na direção oposta, como *flow unicast*, já que o autenticador fica a saber o endereço MAC do *Supplicant* ao receber o pacote 802.1X. Quando o *host* se liga começa a autenticação ao enviar uma mensagem *Start*. O *host* passa para o estado *Pending*. No estado *Pending* todo o tráfego para o *host* ainda é ignorado, mas o *host* já está à espera da confirmação de autenticação do autenticador para o controlador. Quando a autenticação é confirmada, o *host* passa para o estado *Authenticated*. Neste estado a aplicação a correr no controlador permite o acesso aos recursos da rede. O controlador verifica que o tráfego está de acordo com as políticas definidas para o *host* em questão. Se uma *flow* tem nos seus campos o endereço MAC de origem e o *incoming port* do *switch* igual aos do *host*, as credenciais de autenticação e a identidade do *host* são atribuídas a esta *flow*. Logo o controlo de acesso é feito de acordo com a credencial da *flow* e não apenas de acordo com os campos da *flow*. Similarmente, se uma *flow* tem nos seus campos o endereço MAC de destino e o *output port* iguais aos de um *host*, a credencial de autorização desse *host* também é atribuída à *flow*.

2.2.4 Conflitos de regras

Em sistemas de controlo de acesso e autenticação que utilizem tecnologia OpenFlow, podemos encontrar situações em que duas regras OpenFlow estão em conflito. Um conflito entre regras surge quando uma nova regra OpenFlow desativa uma regra existente que estava ativa, ou então, ativa uma regra que estava desativa.

No sistema FortNox [13] foi desenvolvido um algoritmo específico para detetar conflitos entre regras. Quando nos deparamos com um conflito e com a dúvida entre aceitar ou rejeitar a nova regra, a decisão tomada depende do tipo de autorização atribuída aos autores das duas regras em conflito. A regra que for introduzida pelo autor com um maior nível de autenticação prevalecerá.

Para detetar um conflito entre regras, o FortNox incorpora uma funcionalidade que efetua a conversão das regras para um tipo de representação próprio definido como *alias reduced rules*, que deteta contradições entre as regras [13]. Uma *alias reduced rule* é apenas uma derivação simples da regra em que o critério de *matching* incorpora também operações *set* e *wildcards*. Este tipo de representação é utilizado para que posteriormente seja feita a análise de conflitos. Na presença de um conflito, o FortNox faz a redução a *alias set rule* da candidata a nova regra e depois efetua testes de validação entre a candidata a nova regra e a regra ativa, na forma de *alias*

reduced rule. Quando surge um conflito, a resolução do mesmo é feita dependendo da prioridade atribuída ao *role* dos agentes que originam as regras, se a prioridade definida para a nova regra for superior à da regra em conflito na tabela agregada, então esta é substituída pela nova regra.

2.3 Modelo de gestão de políticas de autenticação e de controlo de acesso

Um modelo de gestão de políticas de autenticação e de controlo de acesso é desenvolvido para definir as políticas que determinam o comportamento da rede. As políticas de autenticação e de controlo de acesso são declaradas como um conjunto de regras. Essas regras consistem em predicados e num conjunto de ações resultantes, que são realizadas para as *matching flows*, as *flows* que respeitam as condições definidas na política [12].

A estruturação da linguagem definida no contexto de um modelo pode ser feita de duas formas distintas. Através da definição de eventos, condições e ações, ou definindo condições e ações, sem a definição de eventos. Algumas linguagens estão assentes no paradigma Evento-Condição-Ação em que em cada regra é definido um evento explícito que faz com que ocorram as ações definidas na regra, dentro das condições definidas nessa mesma regra.

O *Role-Based Access Control* está na base de alguns modelos e linguagens, através da utilização de *roles* ou perfis para interligar os *hosts* e as permissões. Cada *host* é associado a um *role* ou perfil e cada perfil pode ter um conjunto de políticas aplicadas. Desta forma a especificação de diferentes perfis impede que, quando as necessidades da empresa se alteram, seja necessário atualizar os privilégios de cada utilizador individualmente. Através de uma linguagem ou modelo assente em *Role-Based Access Control*, os perfis podem ser facilmente criados, modificados ou eliminados consoante a alteração do paradigma da rede.

2.3.1 OrBAC (Organization Based Access Control)

OrBAC é um modelo de controlo de acesso desenvolvido para permitir a definição de políticas de segurança independentemente da forma como estas são implementadas [14]. Esta característica fez com que este modelo tivesse sido escolhido como base, para o desenvolvimento da solução desta dissertação, cuja implementação foi pensada no âmbito do *Software Defined Networking*. Os três conceitos básicos do OrBAC são *Subject*, *Action* e *Object* (sujeito, ação e objeto, respetivamente). Estes conceitos permitem a construção das políticas, dado que, uma política no OrBAC determina que um sujeito pode ou não realizar uma ação sobre um objeto [14]. De notar que, cada política é definida no contexto de uma organização, ou seja, neste caso as políticas estão associadas a uma organização que é a empresa cuja rede irá incorporar estes mecanismos de controlo de acesso.

O OrBAC adiciona também os conceitos de *Role*, *Activity* e *View* (cargo/função, atividade, vista). Um *Role* é composto por um conjunto de *Subjects* sobre os quais as mesmas políticas se aplicam. Da mesma forma, uma *Activity* representa um conjunto de ações. Quando se pretende agrupar um conjunto de objetos, esse conjunto designa-se *View*. Os conceitos que compõem o modelo OrBAC estão ilustrados na Figura 6.

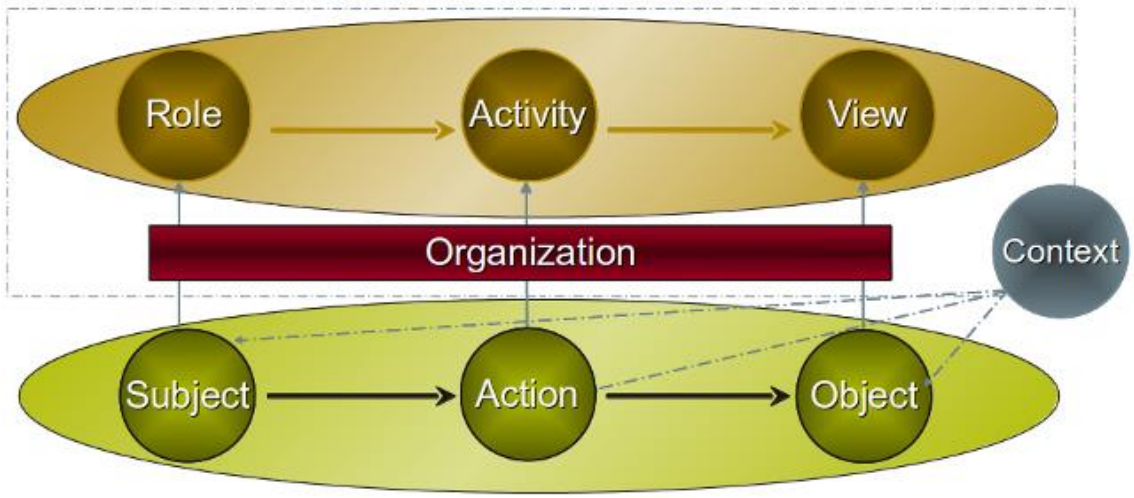

Figura 6: Conceitos que constituem o modelo OrBAC [14]

O modelo OrBAC utiliza um formalismo baseado em lógica de primeira ordem com negação e as suas políticas podem ser expressas através de uma notação similar à sintaxe da linguagem Prolog. Através de uma notação desse tipo podemos exprimir uma permissão da seguinte forma:

- *Permission(enterprise, role, activity, view, context)*

Considerando, por exemplo, uma política específica, associada ao *Role network administrator*, à *Activity administrative access* e à *View servers* obtemos a representação:

- *Permission(enterprise, network administrator, administrative access, servers, default context)*

Esta política significa que na organização *enterprise*, o *role Network Administrator* tem acesso administrativo sobre os servidores, em qualquer contexto.

As políticas OrBAC dividem-se em três categorias: permissões, proibições e obrigações. Com a mesma notação podemos proibir utilizadores de efetuar determinada ação ou obrigar um utilizador a desempenhar uma ação num determinado contexto.

No modelo OrBAC as entidades referidas anteriormente: *Subject*, *Action*, *Object*, são consideradas entidades concretas. Que estão associadas a sujeitos, ações e objetos concretos. As entidades *Role*, *Activity* e *View* denominam-se entidades abstratas. As entidades concretas podem ser associadas a entidades abstratas: um *Subject* associado a um *Role*, uma *Action* associada a uma *Activity* e um *Object* associado a uma *View*. Para definir estas associações são utilizados predicados com este objetivo.

- *empower(org, subject, role)*
- *consider(org, action, activity)*
- *use(org, object, view)*

O predicado *empower* atribui um *Subject* a um *Role*; o predicado *Consider* permite considerar uma *Action* como pertencente a uma *Activity*; e o predicado *Use* indica que um objeto é utilizado numa *View*.

O OrBAC introduz também o conceito de *Context*. *Contexts* são definidos a partir de regras lógicas. Essas regras lógicas expressam a condição que tem que ser verdadeira para que um *Context* esteja ativo [14]. No fundo, o *Context* define uma condição para que determinada política seja aplicada. Existe também a possibilidade de combinar vários *Contexts* para expressar um *Context* conjuntivo ou disjuntivo.

3 Gestor de políticas de autorização e de acesso numa rede empresarial

O processo de desenvolvimento de um gestor de políticas de autorização e de acesso começou com a criação de perfis e políticas, que se podem utilizar no domínio de uma rede empresarial. Para concluir um sistema de controlo de acesso completo passou-se por várias etapas. Começou na definição dos perfis e políticas a implementar. Passou pela formalização desses perfis e políticas, através de uma linguagem/modelo de controlo de acesso e finalizou na implementação das políticas numa rede, a partir de uma aplicação SDN.

3.1 Modelo de controlo de acesso de uma rede empresarial (perfis e políticas)

Um controlador SDN introduz uma grande facilidade em alterar o comportamento de uma rede, através da alteração do código das aplicações SDN que correm no mesmo. Isto permite uma maior diversidade na implementação de um modelo de controlo de acesso numa rede empresarial do que numa rede tradicional. A diversidade provém da possibilidade de se desenvolverem soluções dedicadas às necessidades de cada empresa. O controlo de acesso torna-se mais fácil, visto que, podemos implementar uma solução centralizada, evitando a necessidade de definir as permissões e as proibições em todos os *switches* da rede utilizando *Access Control Lists*.

Esta simplificação do controlo de acesso é muito interessante, visto que, no mundo empresarial atual cada empresa tem as suas necessidades. Necessidades que dependem de vários fatores como por exemplo: o modelo de negócio da mesma, o produto ou serviço que disponibiliza e a indústria em que se insere. As empresas atuais sofrem alterações constantes na sua organização e nas suas infraestruturas de rede e existem empresas de várias classes em termos de dimensão. Nalgumas empresas pode ser aplicado um modelo de controlo de acesso assente num número reduzido de perfis e de políticas e numa empresa de grande dimensão a criação de muitos perfis acaba por ser necessária para realizar o controlo de acesso em toda a rede. A flexibilidade que o controlo de acesso baseado num controlador SDN introduz, permite adaptar um modelo de controlo de acesso como o definido desta dissertação, para empresas que apresentem vários tipos de dimensão e de estrutura. Quando se depara com uma alteração na estrutura de uma grande empresa, como por exemplo, a introdução de novos colaboradores ou até mesmo de novos departamentos, este tipo de solução é interessante na medida em que se pode de forma simples e rápida criar um novo perfil cujas políticas serão implementadas com recurso ao controlador.

A grande diversidade que se encontra hoje em dia no que diz respeito à estrutura que as empresas podem adotar fez com que, na solução desta dissertação, se optasse por desenvolver um modelo simples no que diz respeito aos perfis criados e às políticas definidas para cada um deles. O conjunto de perfis contidos neste modelo não foi pensado com o objetivo de dizer quais são os perfis que devem existir numa empresa ou na generalidade das empresas. Para efetuar a definição dos perfis que vão de encontro às necessidades de uma empresa é necessário um conhecimento profundo da realidade da mesma e da sua estrutura. O modelo aqui apresentado foi desenvolvido de um ponto de vista académico, para ilustrar as capacidades do *Software Defined Networking*, do protocolo OpenFlow e principalmente para demonstrar a utilidade de interligar modelos de

controlo de acesso tradicionais com o controlador HP VAN SDN. A solução apresentada foi desenvolvida tendo em conta aquilo que pode ser considerada uma rede empresarial típica, apesar de esse conceito hoje em dia ser difícil de precisar. Nesta solução apresentamos vários perfis de autenticação para que o controlo de acesso seja fino e eficiente. Considerámos um conjunto de perfis que permitem ilustrar funcionalidades simples de controlo de acesso, implementadas com recurso a um controlador SDN.

Existem vários modelos de controlo de acesso, alguns deles já disponíveis há vários anos. A solução aqui apresentada partiu de um tipo específico de controlo de acesso que é o controlo de acesso baseado em *Roles* (como são referidos no modelo OrBAC) e que neste modelo se denomina de perfis. Partiu-se da ideia de incorporar o modelo OrBAC nesta própria solução que engloba as várias fases da criação de um modelo de controlo de acesso, desde da criação de perfis à implementação de políticas no domínio do *Software Defined Networking*. A vantagem desta solução é a possibilidade de um gestor que não esteja familiarizado com tecnologia SDN poder criar um conjunto de políticas e perfis com o auxílio da ferramenta MotOrBAC para que esses perfis e políticas sejam traduzidos para uma aplicação SDN.

Com base neste conceito esta solução estruturou-se em várias fases. A primeira fase é referente à criação de um modelo teórico de controlo de acesso, baseado no modelo OrBAC, através da definição de um conjunto de políticas e perfis. Políticas e perfis esses que foram formalizados através do *software* MotOrBAC.

Tendo em conta que o MotOrBAC não gera um ficheiro de políticas que seja diretamente interpretado em JAVA através da aplicação SDN que se desenvolveu, na segunda fase do projeto é necessária uma compilação da informação do ficheiro gerado pelo MotOrBAC para informação interpretável pela aplicação. Sendo este um mero problema de compilação e que sai fora do âmbito da configuração e gestão de redes, não se efetuou o compilador fazendo-se uma simplificação de processos que será mais detalhada em 3.3. Numa terceira fase efetuou-se a implementação dos perfis e das políticas através da criação de uma aplicação SDN com o controlador HP VAN SDN.

Esta estrutura delineada para esta dissertação estabeleceu então como primeiro objetivo a criação teórica de um conjunto de políticas e perfis para posteriormente se implementar na aplicação SDN.

O primeiro perfil criado foi um dos que se considerou como sendo um dos mais evidentes e que se denominou de *Network Administrator*. O *Network Administrator* é o gestor da rede de uma organização, que mantém a rede operacional, e monitoriza as funções da mesma. A instalação, a manutenção e o *upgrade* do *software* e do *hardware* necessário para o funcionamento da rede é da sua responsabilidade [15]. Tendo em conta as funções que um *Network Administrator* deve desempenhar, quem pertence a este perfil acaba por possuir um elevado nível de privilégios e de permissões.

O segundo perfil é um dos que, tal como o primeiro, deve ser considerado para a maioria das empresas. Engloba os utilizadores que não são colaboradores diretos da empresa, os visitantes ou *Guests*. O perfil *Guest* é o que apresenta um nível mais baixo de permissões e que tem em geral os seus acessos mais restringidos. No fundo o *Guest* não pode comunicar com a rede interna da empresa. A permissão que lhe é atribuída é apenas o acesso à internet. Todos os outros acessos, a bases de dados internas, a servidores internos, ou a qualquer outra máquina da empresa está bloqueado.

Para definir as permissões dos utilizadores que são verdadeiramente colaboradores da empresa, foi criado outro perfil. Um perfil onde se inserem os funcionários comuns da empresa, que não requerem permissões muito específicas. Visto que, em muitas empresas se tem funcionários que

desenvolvem a sua atividade precisando apenas de recorrer a um computador com acesso à internet e não necessitando de muitos mais recursos no seu trabalho diário, criou-se um perfil específico para eles. Para se transmitir a ideia de que este perfil pode ser criado e atribuído a vários tipos de utilizadores, que podem existir nos mais variados tipos de empresas, manteve-se uma denominação genérica. Denominou-se este perfil de funcionários da empresa simplesmente de *Employee*, visto que, este é um termo genérico que pode englobar colaboradores de uma empresa que à partida não se sabe a que área pertence. Neste contexto de gestão de uma rede o foco está num modelo de controlo de acesso que possa ser aplicado a vários tipos de empresas e este perfil foi criado com esse objetivo. Os perfis *Employee* e *Guest* diferenciam-se na questão dos acessos internos (acessos a recursos da empresa). Enquanto um *Employee* tem acesso a algumas máquinas da empresa, esse acesso é bloqueado para um *Guest*.

Para abordar problemas de gestão de controlo de acesso um pouco mais específicos foram desenvolvidos dois perfis adicionais. O perfil *Accounting* e o perfil *Research & Development*, para ilustrar exemplos de permissões ou negações de acesso que podem surgir em muitas empresas. Quando se encontra numa empresa um departamento de *Accounting* ou *Research & Development* pode ser necessária a criação de um perfil próprio para cada um dos colaboradores desses departamentos. Isto porque, tanto os colaboradores da área da contabilidade como os que trabalham no desenvolvimento de novas soluções e produtos no seio de uma empresa, regra geral, terão acessos próprios aos recursos da rede. A especificidade destes perfis pode ser entendida mais facilmente através das políticas que se criaram para cada um deles. De notar que, foram criados estes dois perfis com base em dois tipos de departamentos que existem em muitas empresas, mais poder-se-ia considerar outros dois departamentos com funções diferentes. Foram estes os escolhidos, dado que nos permitem definir políticas que exemplificam algumas das aplicações interessantes do SDN em termos de controlo de acesso.

Nesse sentido, e tendo em conta estes cinco perfis tomados como exemplo para um modelo de controlo de acesso, decidiu-se organizar as políticas a aplicar em quatro classes. Isto serve também para estender um pouco o alcance desta solução, visto que o controlo de acesso por si só não permitiria criar uma lista muito extensa de políticas sem que a utilidade e o conceito por detrás delas se tornassem repetitivos. Apesar de ser um problema de grande importância, o controlo de acesso não permite uma grande variedade de políticas porque a sua função e aplicação resume-se à permissão ou proibição do acesso a uma máquina ou a um conjunto de máquinas. Para enriquecer este modelo e para dar a conhecer outras possibilidades introduzidas pelo SDN e pelo protocolo OpenFlow consideraram-se então algumas políticas fora do domínio tradicional do controlo de acesso.

As políticas foram estruturadas em classes para demonstrar que podem ser definidas políticas assentes em diferentes características do protocolo OpenFlow, do controlador HP VAN SDN e baseadas em diferentes finalidades.

A primeira classe de políticas que foi criada, define a permissão ou negação da utilização de um determinado serviço da rede empresarial. Por exemplo, a utilização de um serviço de correio eletrónico, de um serviço de vídeo chamada, ou de transferência de ficheiros. Para além do controlo do acesso a uma determinada máquina ou recurso da rede, as políticas pertencentes a esta classe definem quais os serviços que os utilizadores de um perfil podem utilizar. Ao gerir a rede de uma empresa pode-se ter de definir, por exemplo, quais os serviços de que os membros do perfil *Accounting* podem usufruir. Pode-se dizer se o perfil *Accounting* pode efetuar transferências de ficheiros e no caso de isso ser permitido, definir também a partir de que máquinas é que isso pode ser feito.

Uma segunda classe de políticas foi definida com o objetivo de garantir qualidade de serviço para alguns perfis. Considerou-se esta classe tendo em conta que se podem criar vários tipos de

políticas relacionadas com QoS utilizando as classes Java que estão à disposição programando sobre o controlador HP VAN SDN. Exemplificando o tipo de políticas que podem existir nesta classe, uma das políticas que definiu permite estipular valores de largura de banda para cada perfil existente na rede.

A terceira classe de políticas é menos abrangente do que a primeira e aborda uma questão mais particular, especificamente relacionada com o acesso à internet que pode ser feito pelas máquinas ligadas à rede empresarial. No fundo, esta classe diz respeito ao bloqueio do acesso a determinados domínios da internet. Para ilustrar o interesse deste tipo de políticas basta pensar que a maior parte dos colaboradores de uma empresa pode verificar uma diminuição da sua produtividade por estar a aceder a *websites* que não são necessários para o desenvolvimento da sua atividade laboral e que funcionam como uma distração.

A quarta classe de políticas engloba políticas que forçam o tráfego de um perfil a ser encaminhado para uma máquina específica antes de chegar ao seu destino. Uma situação que pode exigir a implementação de uma política destas é a existência de vários servidores HTTP numa rede empresarial. Neste caso, pode ser criada uma política que faça com que todo o tráfego que tem como destino um servidor HTTP passe num *load balancer* antes de chegar ao servidor, para ser realizado um balanceamento de carga.

Para demonstrar a aplicabilidade destas classes de políticas num contexto concreto, desenvolvido para exemplificar o funcionamento do modelo de controlo de acesso, criou-se uma topologia constituída por este conjunto de máquinas.

- Servidor HTTP
- Servidor FTP
- Servidor de aplicação Web da empresa
- Servidor de impressão
- Servidor de *Instant messaging*
- Servidor SMTP
- Servidor HTTP adicional para o departamento *Research & Development*
- Base de dados para o departamento *Accounting*

A topologia simulada no Mininet encontra-se ilustrada na Figura 7.

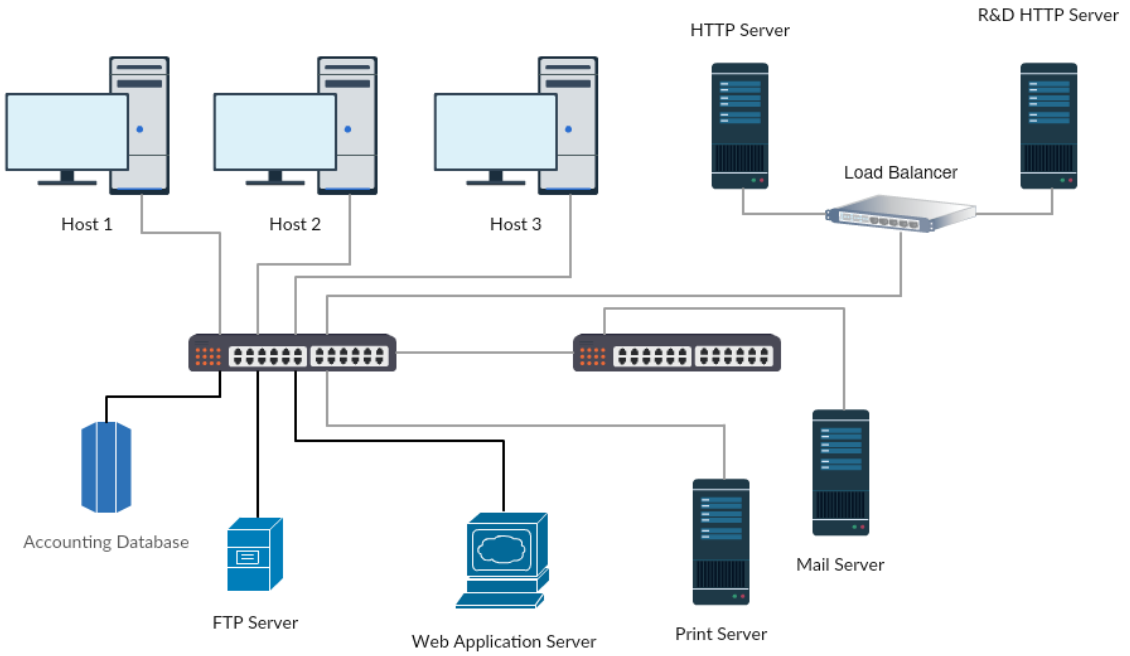

Figura 7: Topologia de Rede Empresarial

3.1.1 Classe de políticas de controlo de acesso a serviços

As classes criadas serviram de base para o desenvolvimento de um conjunto de políticas. Dentro da primeira classe, foram criadas políticas de controlo de acesso a serviços. Estas políticas têm por base o controlo da utilização dos serviços associados a protocolos correspondentes à camada aplicação do modelo OSI. Como por exemplo os protocolos HTTP, FTP, SMTP, SIP, entre outros. Relativamente a cada um destes protocolos é feito o controlo da utilização dos serviços relacionados com os protocolos e também o controlo do acesso aos vários servidores que disponibilizam esses serviços. Para exemplificar esta metodologia, podemos adiantar que no modelo desta dissertação, para um utilizador pertencente ao perfil *Guest* é permitida a comunicação através do protocolo HTTP para a internet mas não o acesso ao servidor HTTP da empresa.

Apresenta-se em seguida uma listagem das políticas de controlo de acesso a serviços para cada perfil. Dentro desta primeira classe de controlo de acesso a serviços, para o perfil *Network Administrator*, onde se inserem os gestores da rede definiram-se as políticas da Tabela 1. Estas políticas foram baseadas na premissa de que numa rede empresarial os gestores da mesma podem ter acesso a todos os serviços, de forma a possuírem todos os privilégios necessários para exercerem as suas funções, tipicamente de gestão e configuração da rede em questão.

Tabela 1: Políticas da classe de controlo de acesso a serviços para o perfil Administrator

Permitido o acesso ao servidor HTTP da empresa
Permitido o tráfego HTTP de e para outras máquinas
Permitido o acesso ao servidor FTP da empresa
Permitido o tráfego FTP de e para outras máquinas
Permitido o acesso ao servidor HTTP do departamento R&D
Permitido o à base de dados do departamento *Accounting*
Permitido o acesso ao servidor de aplicações da empresa
Permitido o acesso ao servidor de impressão da empresa
Permitido o acesso ao servidor de e-mail da empresa
Permitida a utilização do serviço de e-mail da empresa
Permitido o acesso ao servidor VoIP da empresa
Permitida a utilização do serviço VoIP da empresa
Permitido a utilização do protocolo SSH (Secure Shell)
Permitida a utilização do protocolo TELNET

O perfil *Guest*, foi criado para ser atribuído aos visitantes que não pertencem à empresa, logo em termos de acessos aos serviços existentes na rede, os utilizadores pertencentes a este perfil têm um acesso muito restringido. Isso pode ser verificado pelo conjunto de políticas da classe de controlo de acesso a serviços atribuídas a este perfil, listadas na Tabela 2. Sumarizando as permissões e os bloqueios de acesso a serviços definidas nas políticas da Tabela 2, o único acesso permitido aos utilizadores do perfil *Guest* é o acesso à internet.

Tabela 2: Políticas da classe de controlo de acesso a serviços para o perfil Guest

Bloqueado o acesso ao servidor HTTP da empresa
Permitido o tráfego HTTP de e para outras máquinas
Bloqueado o acesso ao servidor FTP da empresa
Bloqueado o tráfego FTP de e para outras máquinas
Bloqueado o acesso ao servidor HTTP do departamento R&D
Bloqueado o acesso administrativo à base de dados do departamento *Accounting*
Bloqueado o acesso ao servidor de aplicações da empresa
Bloqueado o acesso ao servidor de impressão da empresa
Bloqueado o acesso ao servidor de e-mail da empresa
Bloqueada a utilização do serviço de e-mail da empresa
Bloqueado o acesso ao servidor VoIP da empresa
Bloqueada a utilização do serviço VoIP da empresa
Bloqueada a utilização do protocolo SSH (Secure Shell)
Bloqueada a utilização do protocolo TELNET

Na conjuntura das características do perfil *Employee*, os seus utilizadores necessitam de acesso aos serviços básicos na rede da empresa mas não apresentam nenhum acesso especial que não esteja presente nos restantes perfis. Isto porque este perfil foi criado para representar os colaboradores da empresa que não têm qualquer especificidade em termos do departamento a que pertencem ou em relação aos serviços que precisam para desempenhar as suas funções. Relativamente a esta classe de políticas da classe de controlo de acesso a serviçes, as permissões e as negações de acesso do perfil *Employee* estão representadas na Tabela 3.

Tabela 3: Políticas da classe de controlo de acesso a serviços para o perfil Employee

Permitido o acesso ao servidor HTTP da empresa
Permitido o tráfego HTTP de e para outras máquinas
Permitido o acesso ao servidor FTP da empresa
Permitido o tráfego FTP de e para outras máquinas
Bloqueado o acesso ao servidor HTTP do departamento R&D
Bloqueado o acesso administrativo à base de dados do departamento *Accounting*
Permitido o acesso ao servidor de aplicações da empresa
Permitido o acesso ao servidor de impressão da empresa
Bloqueado o acesso ao servidor de e-mail da empresa
Permitida a utilização do serviço de e-mail da empresa
Bloqueado o acesso ao servidor VoIP da empresa
Permitida a utilização do serviço VoIP da empresa
Bloqueada a utilização do protocolo SSH (Secure Shell)
Bloqueada a utilização do protocolo TELNET

Na área empresarial, muitas são as organizações que na sua estrutura, apresentam uma secção constituída por colaboradores que desenvolvem a sua atividade no domínio da contabilidade e das finanças. Logo considerou-se no modelo a inclusão de um perfil referente a esse tipo de profissionais. Esse perfil foi pensado, tendo como objetivo, controlar a troca de informação financeira que não deve chegar a profissionais de outras áreas e departamentos. Esse objetivo traduz-se no modelo no bloqueio do acesso dos utilizadores de outros perfis aos do perfil *Accounting* e vice-versa. Para tentar assegurar a segurança de dados que contenham informação de teor financeiro, bloqueou-se também o acesso a uma base de dados da contabilidade que se incorporou na topologia de rede. Esta é a principal distinção a sublinhar quando se comparam as políticas já apresentadas (referentes a outros perfis) com as políticas apresentadas para o perfil *Accounting* na Tabela 4.

Tabela 4: Políticas da classe de controlo de acesso a serviços para o perfil Accounting

Permitido o acesso ao servidor HTTP da empresa
Permitido o tráfego HTTP de e para outras máquinas
Permitido o acesso ao servidor FTP da empresa
Permitido o tráfego FTP de e para outras máquinas
Bloqueado o acesso ao servidor HTTP do departamento R&D
Permitido o acesso administrativo à base de dados do departamento *Accounting*
Permitido o acesso ao servidor de aplicações da empresa
Permitido o acesso ao servidor de impressão da empresa
Bloqueado o acesso ao servidor de e-mail da empresa
Permitida a utilização do serviço de e-mail da empresa
Bloqueado o acesso ao servidor VoIP da empresa
Permitida a utilização do serviço VoIP da empresa
Bloqueada a utilização do protocolo SSH (Secure Shell)
Bloqueada a utilização do protocolo TELNET

O outro perfil baseado num departamento específico existente em várias empresas que se decidiu adicionar ao modelo foi o denominado *Research & Development* (R&D). Este perfil foi incorporado por dois motivos: porque é comum se encontrar um departamento correspondente a este tipo de perfil numa empresa e porque permitiu adicionar algumas políticas diferentes das que já se tínham criado para os perfis anteriores. Nomeadamente no âmbito nesta classe de políticas de controlo de acesso a serviços. Exemplificando e assumindo que se vai implementar o modelo numa empresa que exerce a sua atividade na área da tecnologia, provavelmente ir-se-á encontrar um departamento de investigação e desenvolvimento de *software*. Neste departamento o acesso a alguns serviços, bloqueados para outros perfis, pode ser necessário. No perfil que se criou para este modelo, a título de exemplo, considerou-se que existe na empresa um servidor HTTP dedicado apenas ao departamento de investigação e desenvolvimento, para desenvolvimento de uma aplicação. Essa foi uma das políticas da classe de controlo de acesso a serviços consideradas para este perfil, em conjunto com as restantes que se apresentam na Tabela 5.

Tabela 5: Políticas da classe de controlo de acesso a serviços para o perfil R&D

Permitido o acesso ao servidor HTTP da empresa
Permitido o tráfego HTTP de e para outras máquinas
Permitido o acesso ao servidor FTP da empresa
Permitido o tráfego FTP de e para outras máquinas
Permitido o acesso ao servidor HTTP do departamento R&D
Bloqueado o acesso administrativo à base de dados do departamento *Accounting*
Permitido o acesso ao servidor de aplicações da empresa
Permitido o acesso ao servidor de impressão da empresa
Bloqueado o acesso ao servidor de e-mail da empresa
Permitida a utilização do serviço de e-mail da empresa
Bloqueado o acesso ao servidor VoIP da empresa
Permitida a utilização do serviço VoIP da empresa
Bloqueada a utilização do protocolo SSH (Secure Shell)
Bloqueada a utilização do protocolo TELNET

Estas políticas estão associadas aos diferentes perfis apresentados. No entanto, no contexto de uma rede empresarial podem também ser criadas algumas políticas independentes dos perfis existentes. No modelo não se consideraram políticas desse tipo e optou-se por basear a solução num *Role Base Access Control* puro, visto que, a organização de políticas por perfis deu a flexibilidade necessária para implementar o controlo de acesso que se considerou adequado. No entanto, não se deve esquecer que se pode encontrar a necessidade de definir uma política de rede, aplicável a vários utilizadores de diferentes perfis. Por exemplo, permitindo a utilização de um determinado serviço para um utilizador do perfil *Accounting* e um utilizador do perfil *Research & Development*. Considerou-se relevante explicar que essa possibilidade foi considerada quando se pensou este modelo. Na mesma linha de raciocínio deve-se ter também em conta que numa solução deste tipo, podem ser criadas políticas transversais a todos os perfis da rede, ou seja, a todos os utilizadores da mesma. Este conceito pode ser entendido e imaginado na forma de um perfil superior, que poderia englobar outros sub-perfis. A flexibilidade é uma das vantagens deste modelo, no âmbito desta classe de políticas ou de qualquer uma das outras classes. Porque uma política pode ser aplicada sobre um perfil, sobre um utilizador em específico, ou sobre todos os utilizadores da rede criando um perfil superior.

3.1.2 Classe de políticas de Qualidade de Serviço

Este modelo foi desenvolvido com base no *Role Based Access Control*, ou seja, criando um conjunto de perfis que permitem organizar o controlo de acesso de uma rede empresarial. No entanto, numa rede empresarial, para além de ser importante construir mecanismos de controlo de acesso eficientes é também interessante fazer uma gestão da largura de banda disponível. A definição de perfis pode também auxiliar na resolução do problema da gestão da largura de banda na medida em que o gestor da rede pode distribuir a largura de banda disponível pelos perfis, consoante as necessidades de cada um destes. A organização dos utilizadores de uma empresa a partir de um grupo de perfis permite a criação de políticas que garantam a qualidade de serviço para cada um destes.

Considerando os perfis *Guest*, *Network Manager*, *Employee*, *Accounting* e *Research & Development* atribuiu-se um valor de largura de banda a cada utilizador consoante o perfil em que este se insere. O que determina a largura de banda atribuída a cada utilizador é o perfil a que este pertence. Optou-se por definir esta distribuição de largura de banda baseada em perfis, dado que, em qualquer empresa onde possa ser utilizado um modelo deste tipo, existe sempre um valor limitado de largura de banda para distribuir por toda a empresa.

Os perfis possuem diferentes requisitos em termos de acessos a serviços tal como foi definido na primeira classe de políticas. No entanto, consoante as funções a desempenhar nem todos os perfis necessitam do mesmo valor de largura de banda. Admitindo que existe um perfil destinado a visitantes como o perfil *Guest* deste modelo, mesmo permitindo o acesso destes utilizadores à internet como é o caso, não se deve atribuir um valor de largura de banda para os *Guests* que possa prejudicar os restantes perfis em que se inserem os membros da empresa. Logo assumiu-se para este perfil Guest o menor valor de largura de banda. O perfil *Network Manager* e o Research & Development englobando os gestores da rede e um possível departamento de Investigação e Desenvolvimento respetivamente, têm a maior parte da largura de banda total disponível. Isto porque as suas funções assim o exigem, sejam elas de configuração e gestão da rede ou de desenvolvimento de aplicações e produtos. Os restantes perfis *Employee* e *Accounting* não desempenhando funções que necessitem de grande parte da largura de banda total apresentam valores inferiores relativamente aos dois perfis referidos anteriormente.

Assumindo que se tem uma largura de banda disponível para distribuir pelos perfis de 1 Gigabit por segundo pode-se atribuir os seguintes valores de largura de banda para cada perfil:

- *Guest* – 50 Mbit/s
- *Network Manager* – 250 Mbit/s
- *Employee* – 100 Mbit/s
- *Accounting* – 100 Mbit/s
- *Research and Development* – 500 Mbit/s

Como na topologia definida se tem apenas um utilizador por perfil então a um *Guest* é atribuído um limite de 50 Mbit/s, a um Network Manager 150 Mbit/s e o mesmo é válido para os restantes perfis.

Para além das políticas de distribuição de largura de banda, dentro desta classe de políticas de Qualidade de Serviço, tem-se também o que se denomina de políticas *Highspeed* que permitem dar aos membros de um determinado perfil, ligação *Highspeed* através da utilização de portas dos *switches* HP que garantem um maior ritmo binário (*bit rate*). Desta forma pode-se ligar uma

máquina a uma das portas dos *switches* que garantem maior ritmo binário e permitir que alguns perfis acedam a essa máquina a partir dessa porta. Para se demonstrar esta possibilidade definiu-se que o perfil *Research & Development* tem uma ligação *Highspeed* ao servidor HTTP.

3.1.3 Classe de políticas de bloqueio de acesso a Websites

A terceira classe de políticas é menos abrangente do que a primeira e aborda uma questão mais particular, especificamente relacionada com o acesso à internet que pode ser feito pelas máquinas pertencentes à rede empresarial que se está a gerir. No fundo, esta classe diz respeito ao bloqueio do acesso a determinados *websites*. Para entender a importância deste tipo de políticas basta pensar num cenário comum. Em muitas empresas, grande parte dos seus funcionários verifica, no seu dia-a-dia, uma diminuição da sua produtividade que é causada pela utilização de determinados *websites.* Tipicamente, muitos dos *websites* que causam uma diminuição de produtividade são *websites* de redes sociais. Esta é apenas uma tendência típica, sendo que, o conjunto de *websites* que impacta de forma negativa a produtividade de um funcionário depende sempre das preferências do mesmo. No fundo pode-se considerar neste modelo que seria interessante bloquear qualquer *website* que não seja necessário para o desenvolvimento da atividade laboral dos membros de um determinado perfil.

Numa rede empresarial pode ser bloqueado o acesso a vários Websites conforme as necessidades da empresa. Existem vários motivos que podem levar o gestor de uma rede a bloquear um determinado domínio. Quer seja por motivos de mera produtividade da empresa quer seja por razões de segurança da rede. Neste modelo considera-se que o bloqueio seria feito na rede de uma empresa para que os trabalhadores sejam mais produtivos e não deixem de fazer o seu trabalho para navegar em *websites* que funcionam como uma distração. Dentro do conjunto de *websites* que tipicamente funcionam como distração, decidiu-se considerar os que se inserem no grupo que foi referido anteriormente, os *websites* de redes sociais. Como tal, bloquearam-se algumas das redes sociais de maior dimensão como Facebook, Youtube, Twitter, Pinterest, Tumblr e Instagram.

Na implementação prática é possível listar os domínios bloqueados para cada perfil, ou seja, os *websites* bloqueados para um certo perfil podem não estar bloqueados para outro perfil. Adicionou-se esta funcionalidade para demonstrar que a mesma pode ser implementada com recurso ao controlador e para ter em consideração que em algumas empresas específicas o gestor da rede pode considerar bloquear uma lista diferente de *websites* para cada perfil.

Detalhando um pouco mais as políticas desta classe, partiu-se do princípio de que o gestor da rede não tem como objetivo controlar a produtividade de um *Guest*, logo não se bloqueou nenhuma das redes sociais referidas para este perfil. O perfil *Network Administrator* é o segundo perfil que não tem bloqueado qualquer *website*, visto que este é o perfil dos próprios gestores da rede e que podem ter acesso a qualquer domínio. Decidiu-se então bloquear para o perfil *Employee* as redes sociais Facebook, Youtube e Twitter e para o perfil *Accounting* as redes sociais Pinterest, Tumblr e Instagram. Esta divisão foi feita apenas para demonstrar a possibilidade de bloquear diferentes domínios para diferentes perfis, como foi referido anteriormente. De notar que este modelo foi desenvolvido de forma a ilustrar as capacidades do mesmo e que as especificidades como os *websites* a bloquear, entre outro tipo de política, dependem muito da rede empresarial em que se vai incorporar esta solução. Em teoria poderia até fazer sentido bloquear o Facebook, Youtube, Twitter, Pinterest, Tumblr e Instagram tanto para o perfil *Employee* como para o *Accounting*. No entanto, ao adicionar esta divisão pode-se constatar a capacidade de bloquear domínios diferentes para perfis diferentes. Para o perfil *Research & Development*, teve-se em consideração uma outra

questão. Quando se partiu da ideia de bloquear os domínios referidos, teoricamente faria sentido bloquear todos estes domínios também para o perfil *Research & Development*. No entanto, colocou-se a hipótese de que o perfil em questão pode corresponder a um departamento de uma empresa de desenvolvimento de *software* que desenvolva várias aplicações. Como por exemplo, aplicações para o Facebook. Nesse caso não pode ser bloqueado o acesso ao Facebook para os programadores que pertencem ao perfil *Research & Development.* Como tal, bloqueou-se para este perfil todas as rede sociais excetuando o Facebook. Hoje em dia as redes sociais são também uma ferramenta de trabalho para muitas empresas, como tal, decidiu-se considerar essa possibilidade.

Para bloquear o acesso aos *websites* referidos consideraram-se todos os domínios listados na Tabela 6. Visto que são estes alguns dos domínios associados a Facebook, Youtube, Twitter, Pinterest, Tumblr e Instagram.

Tabela 6: Domínios Bloqueados

Website	**Domínios Bloqueados**
Facebook	www.facebook.com facebook.com login.facebook.com www.login.facebook.com fbcdn.net www.fbcdn.net fbcdn.com www.fbcdn.com static.ak.fbcdn.net static.ak.connect.facebook.com connect.facebook.net www.connect.facebook.net apps.facebook.com
Youtube	www.youtube.com www.googlevideo.com www.ytimg.com
Twitter	www.twitter.com
Pinterest	www.pinterest.com
Tumblr	www.tumblr.com
Instagram	photos-f.ak.instagram.com scontent-b.cdninstagram.com distilleryimage4.ak.instagram.com scontent-a.cdninstagram.com images.ak.instagram.com photos-h.ak.instagram.com/hphotos-ak-prn scontent-a.cdninstagram.com/hphotos-frc photos-e.ak.instagram.com

Na solução que se desenvolveu teve-se como objetivo a implementação de mecanismos existentes no modelo OrBAC no contexto do *Software Defined Networking*. Por isso utilizou-se o conceito de *Context* existente no OrBAC, mais especificamente o conceito de *Temporal Context*. No modelo OrBAC o *Context* foi introduzido para expressar regras dinâmicas, no caso do *Temporal Context* para expressar regras que só são aplicáveis durante um período de tempo. Utilizou-se então este conceito para fazer com que os bloqueios de domínios fossem condicionais temporalmente. Em termos práticos adicionou-se mais uma condição às políticas de bloqueio de domínios. Para além da dependência do perfil em questão, adicionou-se uma dependência da hora atual. Mesmo que seja considerado pelos gestores da rede que faz sentido bloquear o acesso a certos domínios na sua rede para efeitos de aumento da produtividade, os gestores da rede podem considerar que isto só deve ser válido durante o horário laboral dos funcionários da empresa. Ou seja, implementaram-se as políticas para que os domínios bloqueados não estejam bloqueados durante o período entre as 12 horas e as 14 horas, assumindo que este seria o horário de almoço de uma determinada empresa.

Como foi referido, numa rede empresarial podem também ser bloqueados domínios que possam representar perigo em termos de segurança ou de privacidade. Não se adicionaram domínios deste tipo, porque do ponto de vista de implementação não acrescentariam nenhum mecanismo diferente à solução. No entanto, é importante salientar que o acesso a este tipo de domínios, tipicamente, deve ser bloqueado para todos os perfis, independentemente da especificidade dos mesmos. Visto que são domínios que podem constituir uma ameaça para qualquer utilizador e comprometer a segurança na rede.

3.1.4 Classe de políticas de reencaminhamento de tráfego

A quarta classe de políticas engloba políticas que não se inserem diretamente na área do controlo de acesso, mas que também se considerou que poderiam ser pertinentes num modelo deste género. Políticas que forçam o tráfego de um perfil a ser encaminhado para uma máquina em específico antes de chegar ao seu destino e que podem ser úteis em várias situações. Uma situação que pode servir como exemplo para implementar uma política deste tipo é a existência de vários servidores HTTP numa rede empresarial. Neste caso, pode ser criada uma política que faça com que todo o tráfego que tem como destino um servidor HTTP passe num *load balancer* antes de chegar ao servidor, para ser feito um balanceamento de carga.

Em termos gerais, pode-se encontrar a necessidade de considerar políticas que forcem o tráfego de um perfil a ser encaminhado por uma máquina específica, se se tiver *middleboxes* na rede. Pode-se ter na rede *middleboxes* como *firewalls*, *Network Address Translators*, *WAN Optimizers*, *Intrusion Detection Systems* ou *load balancers* e pode ser necessário encaminhar certos tipos de tráfego para *middleboxes* como estas consoante as necessidades da empresa e conforme a rede.

Dentro de todas estas possibilidades que existem para políticas deste tipo, na topologia de rede que se utilizou, decidiu-se incluir um *load balancer* para exemplificar esta classe de políticas. Definiu-se uma política que faz com que o tráfego que tem como destino os servidores HTTP da empresa passe em primeiro lugar no *load balancer*. Ao se analisar uma topologia como a que se usou para testar as políticas, contendo apenas dois servidores HTTP, a necessidade de fazer balanceamento de carga não é evidente. No entanto, esta topologia foi considerada apenas para efeitos de teste e num cenário real poder-se-ia encontrar uma empresa com uma rede com um grande número de servidores e a existência de um *load balancer* para balanceamento de carga seria à partida necessária.

Assumindo que a empresa cuja rede se estaria a gerir seria de grande dimensão, provavelmente ter-se-iam muitos pedidos para o servidor HTTP e nesse caso seria necessário aumentar o número de servidores na rede e repartir o tráfego por esses servidores de forma a não congestionar o serviço e a aumentar também a redundância.

Os outros tipos de *middleboxes* poderiam ou não ser incluídos numa rede empresarial. Em relação à existência de uma *firewall*, por exemplo, sendo esta uma rede definida por *software*, a firewall pode ser implementada numa aplicação própria podendo não existir uma máquina específica que funcione como firewall à parte do controlador. Ou seja, assim como foi criada a aplicação de gestão de políticas desta dissertação, também poderia ser criada uma aplicação *firewall*. Por este motivo não se considerou uma *firewall* numa máquina própria e a necessidade de reencaminhar o tráfego para esta acaba assim por não surgir. Para além disso, o adicionar de mais *middleboxes* não acrescentaria muito a esta solução, na medida em que o mecanismo por detrás do encaminhamento do tráfego para qualquer uma das *middleboxes* referidas seria praticamente o mesmo. A aplicação deste tipo de políticas vai variar sempre consoante a rede onde esta solução for incorporada.

3.2 Formalização das políticas através do modelo OrBAC

As políticas criadas teoricamente na primeira fase de desenvolvimento da solução foram introduzidas no editor de políticas MotOrBAC [16], para que fosse feita uma simulação das políticas em questão. Como se pode constatar na Figura 8, o MotOrBAC permite a criação das várias entidades do modelo OrBAC, tanto as concretas, como as abstratas. Nomeadamente *Subjects*, *Actions* e *Objects* e *Roles*, *Activities*, e *Views*.

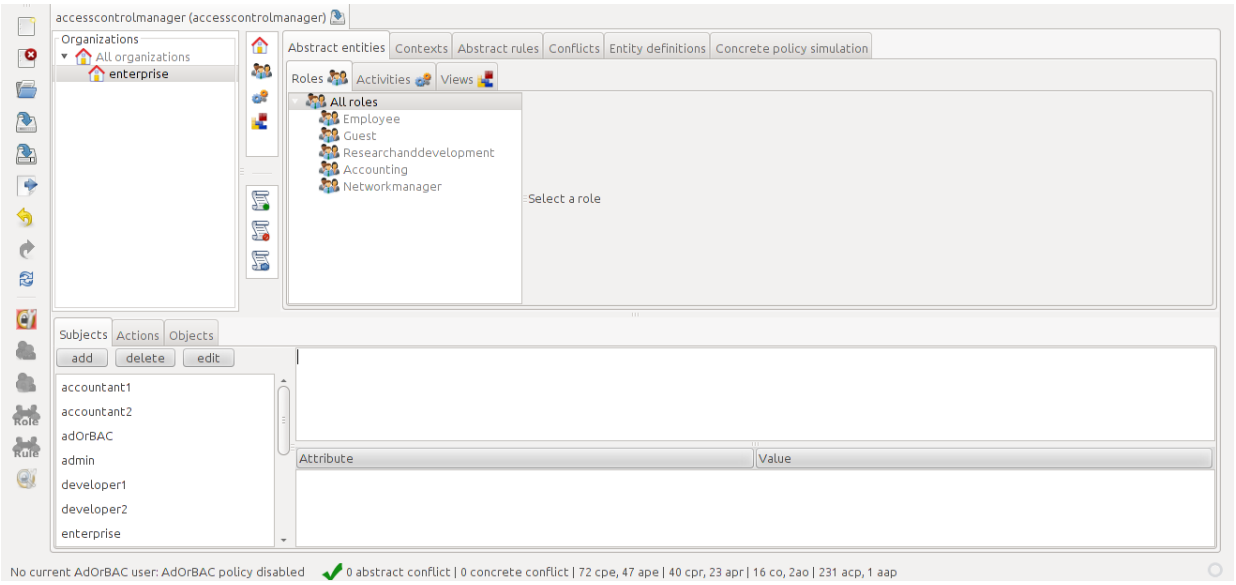

Figura 8: Políticas MotOrBAC

Quando são criadas políticas num modelo com vários perfis, por vezes surgem conflitos entre algumas destas. Através dos mecanismos de resolução de conflitos entre regras conseguiu-se no final não obter nenhum conflito entre as regras de controlo de acesso criadas, como se pode verificar na Figura 9.

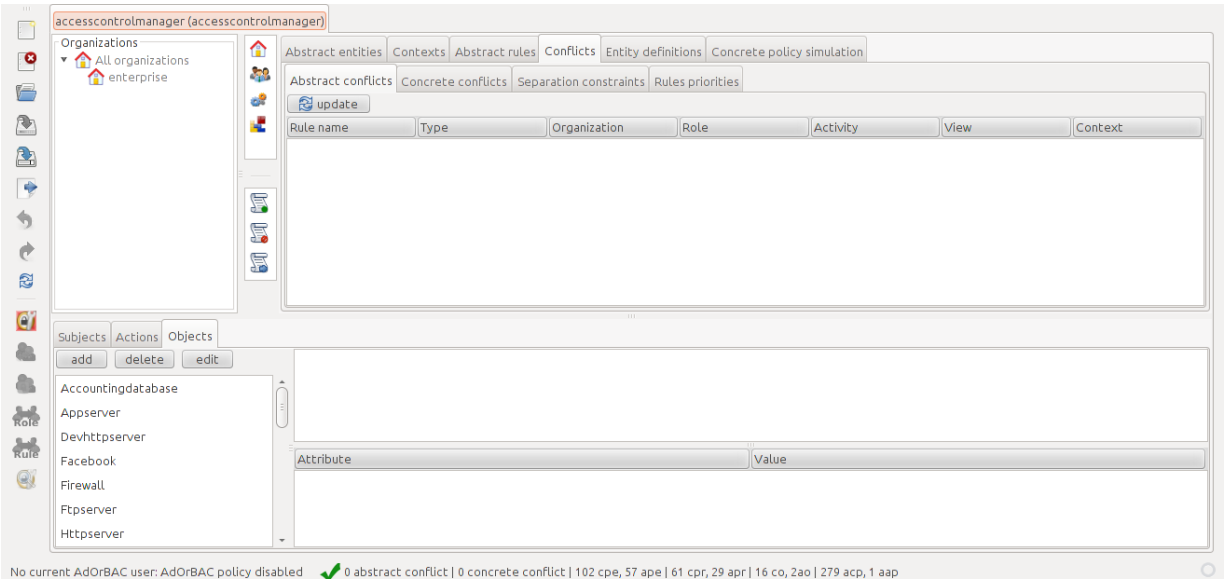

Figura 9: Conflitos no MotOrBAC

No entanto, as regras criadas acabaram por não gerar um grande número de conflitos entre elas dado que foram pensadas cuidadosamente na primeira fase de definição teórica de perfis e políticas. Na ocorrência de um conflito, pode-se verificar na Figura 10 que o MotorBAC apresenta várias possibilidades para permitir resolver esses conflitos. Entre essas possibilidades, tem-se as restrições de separação que permitem separar um *role* de um outro *role*, uma *Activity* de outra *Activity* ou uma *View* de outra *View*, para que se possa garantir, por exemplo, que um *Subject* pertencente ao *Role* Accounting não possa pertencer ao *Role Research & Development*. Outra das soluções para resolver conflitos que o MotOrBAC apresenta é a alteração do nível de prioridade de uma das regras em conflito. Ou seja, fazer com que uma regra seja prioritária em relação à outra.

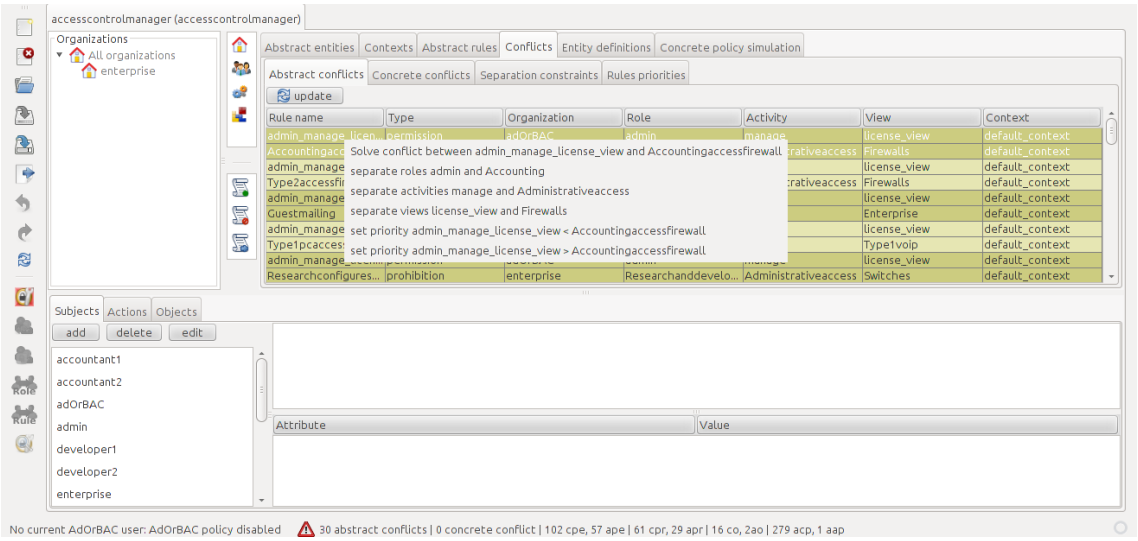

Figura 10: Resolução de conflitos no MotOrBAC

Ainda outra das capacidades que o MotOrBAC possui, que fez com que fosse utilizado na criação destas políticas é a simulação de políticas. Quando é feita a simulação de políticas no MotOrBAC

é possível definir a data da simulação num relógio com essa finalidade, para assim verificar o comportamento de regras que possuam contextos temporais definidos. Por exemplo, nesta solução composta por um conjunto de políticas de controlo de acesso numa rede empresarial, foi criada uma regra que apenas permite aos funcionários pertencentes ao perfil *Employee* aceder a domínios bloqueados na empresa (como o domínio www.facebook.com) entre as 12 horas e as 14 horas (assumindo esta como sendo a hora de almoço). Colocando a data de simulação do MotOrBAC entre as 12 horas e as 14 horas verificou-se que a regra se encontrava ativa como seria de esperar tendo em conta o contexto que foi criado. Quando a data de simulação utilizada estava fora do período entre as 12 horas e as 14 horas, a regra estava inativa como se pode constatar na Figura 11.

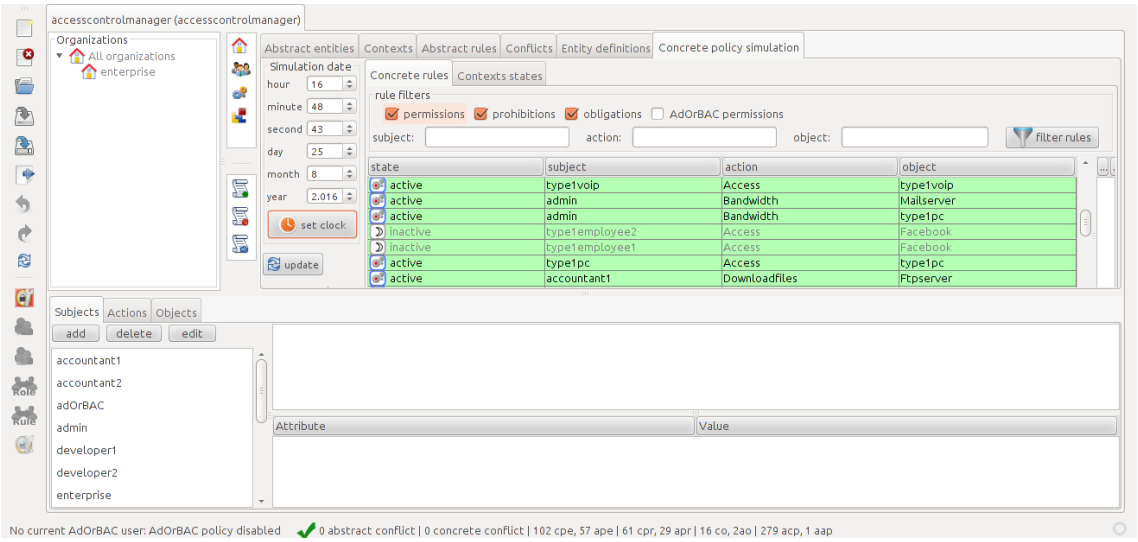

Figura 11: Simulação concreta de políticas no MotOrBAC

Após a simulação das políticas, o MotOrbac foi utilizado para gerar um ficheiro XML, que contivesse assim todas as políticas criadas para gerir a rede SDN. Tendo sido a nossa aplicação SDN implementada em Java não realizou a leitura dos perfis e das políticas a partir do ficheiro XML. No entanto, desenvolvendo um compilador poderia ser feita uma tradução entre as políticas definidas com o modelo OrBAC e a implementação das mesmas em SDN. A simplificação que foi feita, foi considerar um ficheiro de texto com as políticas criadas no MotOrBAC que a aplicação SDN consegue ler. Assume-se que este ficheiro de texto é o resultado da compilação do ficheiro XML gerado pelo MotOrbac.

Traduzindo os perfis apresentados anteriormente para o modelo OrBAC criaram-se os roles: *Guest*, *Network Manager*, *R&D*, *Accounting* e *Employee*. O modelo apresenta então os conjuntos de *Roles*, *Activities* e *Views* das Tabelas 7, 8 e 9 respetivamente.

Tabela 7: Roles do modelo OrBAC

Role
Guest
Network Manager
R&D
Accounting
Employee

Para demonstrar a capacidade do MotOrBAC de definir não só *Roles* principais mas também *sub-roles* foram criados dois *sub-roles* do *Role Employee*. Considerando uma distinção que por vezes pode ser feita nas empresas, a distinção entre os utilizadores que utilizam terminais VoIP dos restantes. Neste caso dentro do *Role Employee* realizou-se essa distinção com a finalidade de bloquear o tráfego entre terminais VoIP e computadores.

No nosso modelo considerámos várias *Activities* que representam os vários tipos de acesso que um utilizador pode ter aos recursos da rede. *Activities* como as da Tabela 8 que representam a utilização de vários serviços através de protocolos comuns como HTTP, FTP, Telnet entre outros.

Tabela 8: Activities do modelo OrBAC

Activity
Access
SSH
Telnet
FTP
Mailing
HTTP
Instant Messaging

No modelo OrBAC os recursos da rede são normalmente considerados como *Objects* e na solução desta dissertação foram agrupados nas *Views* da Tabela 9.

Tabela 9: Views do modelo OrBAC

View
Application Servers
Database Servers
Print Servers
Mail Servers
FTP Servers
HTTP Servers
Development HTTP Servers
Instant Messaging Servers
Firewalls
Switches
Guest
Network Manager
R&D
Accounting
Employee
Blocked websites
External Resources
Enterprise

As políticas criadas com recurso ao MotOrBAC, foram divididas nas três categorias existentes na estrutura do OrBAC. Sendo essas categorias representadas nas tabelas seguintes. Categorias de permissões, proibições e obrigações representadas nas Tabelas 10, 11 e 12 respetivamente.

Tabela 10: Permissões do modelo OrBAC

Role	Action	Object	Context
Network Manager	Access	HTTP Servers	Default context
R&D	Access	HTTP Servers	Default context
Accounting	Access	HTTP Servers	Default context
Employee	Access	HTTP Servers	Default context
Network Manager	HTTP	Enterprise	Default context
R&D	HTTP	Enterprise	Default context
Accounting	HTTP	Enterprise	Default context
Employee	HTTP	Enterprise	Default context
Guest	HTTP	Enterprise	Default context
Network Manager	Access	Development HTTP Servers	Default context
R&D	Access	Development HTTP Servers	Default context
Accounting	Access	Database Servers	Default context
Network Manager	Access	Database Servers	Default context
Network Manager	Access	FTP Servers	Default context
R&D	Access	FTP Servers	Default context
Accounting	Access	FTP Servers	Default context
Employee	Access	FTP Servers	Default context
Network Manager	FTP	Enterprise	Default context
R&D	FTP	Enterprise	Default context
Accounting	FTP	Enterprise	Default context

Role	Action	Object	Context
Employee	FTP	Enterprise	Default context
Network Manager	Access	Application Servers	Default context
R&D	Access	Application Servers	Default context
Accounting	Access	Application Servers	Default context
Employee	Access	Application Servers	Default context
Network Manager	Access	Print Servers	Default context
R&D	Access	Print Servers	Default context
Accounting	Access	Print Servers	Default context
Employee	Access	Print Servers	Default context
Network Manager	Access	Firewalls	Default context
Network Manager	Access	Switches	Default context
Employee (PC User)	Access	Employee (PC User)	Default context
Employee (VoIP User)	Access	Employee (VoIP User)	Default context
Employee	Access	External Resources	Hour ≥12 & Hour ≤14
Guest	Access	External Resources	Default Context
Network Manager	Access	External Resources	Default context
R&D	Access	External Resources	Default context
Accounting	Access	External Resources	Default context
Network Manager	Access	Instant Messaging Servers	Default context
Network Manager	Instant Messaging	Network Manager	Default context

Role	Action	Object	Context
R&D	Instant Messaging	R&D	Default context
Accounting	Instant Messaging	Accounting	Default context
Employee	Instant Messaging	Employee	Default context
Network Manager	Access	Mail Servers	Default context
Network Manager	Mailing	Enterprise	Default context
R&D	Mailing	Enterprise	Default context
Accounting	Mailing	Accounting	Default context
Employee	Mailing	Employee	Default context
Network Manager	Bandwidth	Guest	Default context
Network Manager	Bandwidth	R&D	Default context
Network Manager	Bandwidth	Accounting	Default context
Network Manager	Bandwidth	Employee	Default context
Network Manager	SSH	Enterprise	Default context
Network Manager	TELNET	Enterprise	Default context

Tabela 11: Proibições do modelo OrBAC

Role	Action	Object	Context
Guest	Access	HTTP Servers	Default Context
Guest	Access	Development HTTP Servers	Default Context
Accounting	Access	Development HTTP Servers	Default Context
Employee	Access	Development HTTP Servers	Default Context
Guest	Access	Database Servers	Default Context
R&D	Access	Database Servers	Default Context
Employee	Access	Database Servers	Default Context
Guest	Access	FTP Servers	Default Context
Guest	FTP	Enterprise	Default Context
Guest	Access	Print Servers	Default Context
Guest	Access	Firewalls	Default Context
R&D	Access	Firewalls	Default Context
Employee	Access	Firewalls	Default Context
Accounting	Access	Firewalls	Default Context
Guest	Access	Switches	Default Context
R&D	Access	Switches	Default Context
Employee	Access	Switches	Default Context
Accounting	Access	Switches	Default Context
Employee (PC User)	Access	Employee (VoIP User)	Default Context
Guest	Access	Instant Messaging Servers	Default Context
R&D	Access	Instant Messaging Servers	Default Context

Role	Action	Object	Context
Accounting	Access	Instant Messaging Servers	Default Context
Employee	Access	Instant Messaging Servers	Default Context
Guest	Instant Messaging	Enterprise	Default Context
Guest	Mailing	Enterprise	Default Context

Tabela 12: Obrigações do modelo OrBAC

Role	Action	Object	Context
Network Manager	Access	Servers	Servers down

3.3 Compilação para obter informação interpretável pela aplicação SDN

O editor de políticas MotOrBAC permite criar três tipos de implementações de políticas: JenaOrbacPolicy, XmlOrbacPolicy e MysqlOrbacPolicy. A implementação JenaOrbacPolicy baseia-se no *framework* Jena [23], um *framework* que permite desenvolver aplicações web semânticas. Fornece bibliotecas Java para ajudar os programadores a desenvolver código que usa RDF [17], RDFa [18], OWL [19] e SPARQL [20] com as recomendações W3C [21]. Inclui um motor de inferência baseado em regras para raciocínio baseado em ontologias OWL e RDFS. A implementação em XmlOrbacPolicy guarda as regras abstratas criadas nos MotOrBAC e as entidades concretas associadas às políticas num ficheiro XML e a implementação MysqlOrbacPolicy é uma extensão da XmlOrbacPolicy que guarda as políticas abstratas num ficheiro XML e guarda as entidades concretas numa base de dados Mysql. Tendo em conta que a aplicação SDN desta dissertação foi desenvolvida em JAVA considerou-se que a melhor implementação que se poderia utilizar ao criar as políticas no MotOrBAC seria a MysqlOrbacPolicy. Com a utilização desta implementação o MotOrBAC guardaria as entidades concretas numa base de dados Mysql e estas seriam facilmente lidas e interpretadas na aplicação JAVA que foi desenvolvida. A leitura seria feita de forma mais imediata a partir de uma base de dados do que a partir de um ficheiro XML. No entanto, quando se formalizaram as políticas no MotOrBAC, por erro no próprio *software*, nunca foi possível exportar as políticas de implementação XML para obter as entidades concretas numa base de dados. Segundo a documentação do projeto MotOrBAC a implementação do tipo MysqlOrbacPolicy era considerada como uma funcionalidade experimental em versões do *software* pouco anteriores à atual que se utilizou. Conclui-se que esta funcionalidade deve estar ainda a ser aperfeiçoada no projeto MotOrBAC e talvez no futuro seja possível utilizá-la para implementar um modelo como este. Não se conseguiram assim criar políticas com a implementação MysqlOrbacPolicy e acabou-

se por utilizar a implementação XmlOrbacPolicy obtendo então um ficheiro XML após a simulação das políticas.

Nesse ficheiro XML todos os perfis e políticas ficaram definidos. Foi depois gerado um ficheiro de texto simplificado, facilitando assim a leitura na aplicação SDN. Este ficheiro foi criado de forma a ser visto como possível resultado do processo de compilação do ficheiro XML.

Analisou-se o ficheiro XML gerado pelo MotOrBAC e traduziu-se a sua informação em informação interpretável pelo controlador HP VAN SDN de forma simples.

As políticas da classe de controlo de acesso a serviços foram traduzidas para um ficheiro de texto de forma a garantir que cada política correspondesse a cinco valores. Sendo esses valores: um número que identifica o perfil em questão, a gama de endereços IP das máquinas a que a política diz respeito, uma máscara que corresponde ao número de endereços IP contidos na gama de endereços, o serviço utilizado ou protocolo e uma ação resultante. No ficheiro de leitura estes valores surgem sempre pela seguinte ordem e respeitando a seguinte sintaxe: Perfil - IP Máscara Serviço Ação.

No ficheiro de leitura, cada linha criada representa uma política diferente. Tomando como exemplo uma política que define que o perfil *Guest* não pode aceder ao servidor HTTP temos no ficheiro de leitura, segundo a sintaxe referida:

- 1 - 10.0.0.4 255.255.255.255 HTTP DROP

Traduzindo aquilo que é a informação do ficheiro XML para informação do ficheiro de leitura, temos que, o perfil *Guest* é representado no ficheiro XML gerado pelo MotOrBAC por <role name="Guest"/>. No ficheiro de leitura representou-se essa informação pelo valor 1 que é o valor que se atribuiu ao perfil *Guest*. Nesta política não se tem uma gama de endereços IP associada à política, mas sim um único endereço 10.0.0.4 que é neste caso o endereço IP do servidor HTTP tal como definimos na topologia. Sendo este um endereço IP único, utilizámos a máscara 255.255.255.255. Neste caso o protocolo utilizado para comunicar com o servidor é o HTTP e que surge especificado diretamente na linha referente a esta política e a palavra-chave DROP foi escolhida para identificar uma proibição, pois a ação é o descarte do pacote. Utilizou-se esta palavra-chave no ficheiro porque as políticas da classe de controlo de acesso a serviços foram traduzidas diretamente para *flows* que foram instaladas nos *switches* da rede de modo a estabelecer os comportamentos definidos. Quando se tem uma negação de acesso, os pacotes cujos campos forem iguais aos dos *match fields* de uma certa entrada, são descartados.

Tal como o perfil *Guest*, todos os outros perfis são representados no ficheiro de leitura por um único valor, representados na Tabela 13.

Tabela 13: Representação dos perfis nos ficheiros XML e de leitura

Representação no ficheiro XML	Representação no ficheiro de leitura
<role name="Guest"/>	1
<role name="Networkmanager"/>	2
<role name="Employee"/>	3
<role name="Accounting"/>	4
<role name="R&D"/>	5

O modelo de controlo de acesso OrBAC apresenta na sua estrutura, os conceitos de *Object* e de *View*, cujos significados já foram referidos. Para a tradução da informação presente no MotOrBAC para um ficheiro de leitura deve-se ter em consideração que uma *View* representa um conjunto de *Objects*. Considere-se a estrutura de cada linha do ficheiro de leitura, em que se têm as palavras-chave: Perfil, IP, Máscara, Serviço e Ação. O IP é uma representação de um *Object* ou de uma *View*. Ou seja, um IP único, representa um *Object* que em termos de implementação se traduz numa máquina em concreto (um servidor FTP, por exemplo). Se na posição da palavra-chave IP estiver uma gama de endereços e não apenas um endereço único, então essa gama de endereços representa o que foi definido no modelo OrBAC e no ficheiro XML como uma *View*. Sendo que uma *View* pode corresponder, por exemplo, a um grupo de máquinas que englobe todos os servidores da empresa ou a todas as máquinas usadas por utilizadores do perfil *Accounting*.

Os conceitos *Objects* e *Views* traduzem-se no ficheiro em endereços IP como os representados na Tabela 14. Nem todos os Objects e Views do modelo estão representados na Tabela 14. Apenas se consideraram alguns para exemplificar o tipo de tradução que se efetuou entre estes conceitos e as suas representações no ficheiro de texto final.

Tabela 14: Representação de Views nos ficheiros XML e de leitura

Representação no ficheiro XML	Representação no ficheiro de leitura
<view name="Ftpservers"/>	10.0.0.7
<view name="Httpservers"/>	10.0.0.4
<view name="Databaseservers"/>	10.0.0.6

Uma *View* é um conjunto de *Objects* e da mesma forma uma *Activity* é um conjunto de *Actions*. O que no modelo desta dissertação, no MotOrBAC, é definido como HTTP (*Activity* ou *Action*) traduz-se no ficheiro de leitura nas duas palavras-chave "Serviço" e "Ação" com as representações de HTTP e NORMAL, respetivamente. Utilizou-se a palavra Normal para definir a permissão do tráfego de um determinado Serviço assim como a expressão DROP para definir

uma negação. Tendo em consideração que se criou também uma classe de políticas de qualidade de serviço, então admitiu-se que a palavra-chave “Ação” também pode ser HIGHSPEED para representar uma regra que define uma ligação com maior largura de banda para o utilizador, serviço e destino em questão.

A Tabela 15 mostra a representação de algumas *Activities* tanto no ficheiro XML como no ficheiro de leitura.

Tabela 15: Representação de Activities nos ficheiros XML e de leitura

Representação no ficheiro XML	**Representação no ficheiro de leitura**
<activity name="FTP Access"/>	FTP
<activity name="HTTP Access"/>	HTTP
<activity name="Access"/>	ICMP

Neste ficheiro de leitura, apenas se listaram as políticas que se traduzem diretamente em *Flows* como é o caso de todas as políticas da classe de controlo de acesso a serviços, a primeira classe, e também de um tipo específico de políticas da segunda classe que são as políticas que atribuem uma ligação de alta velocidade a um utilizador de um determinado perfil.

Algumas políticas, pelo facto de não se traduzirem diretamente em *Flows*, sendo antes aplicadas com recurso a outros mecanismos do protocolo OpenFlow e da linguagem, não foram traduzidas para este mesmo ficheiro. Um dos tipos de políticas que não foi implementado utilizando *flows* e que por isso não consta neste ficheiro de leitura diz respeito às políticas da classe de bloqueio de acesso a *websites*. Estas políticas foram listadas num outro ficheiro de leitura, em que constam os *websites* a bloquear para cada perfil.

Na classe de políticas de qualidade de serviço, também existem políticas de distribuição da largura de banda disponível, pelos vários perfis. Estas políticas também não foram implementadas utilizando *flows* daí não constarem no primeiro ficheiro de leitura.

As políticas de redirecção de tráfego, ou seja, as políticas da classe de reencaminhamento de tráfego foram implementadas diretamente na aplicação de gestão de controlo de acesso que se desenvolveu e não foram assim lidas de nenhum ficheiro.

3.4 Tradução dos conceitos dos ficheiros de leitura e implementação de políticas

Quando se atingiu uma fase no desenvolvimento do modelo em que os ficheiros de leitura tinham sido preparados, para que a sua informação fosse lida na aplicação SDN, partiu-se para a fase de implementação das políticas. Para realizar a implementação necessitou-se de definir, por exemplo, o que significa permitir o tráfego HTTP para um utilizador do perfil *Guest*. Neste exemplo concreto deve-se criar uma *flow* em que um dos *match fields* é o porto correspondente ao protocolo HTTP (o porto 80) e a ação associada à *flow* é do tipo *Action Type Output*.

Para demonstrar a tradução de conceitos do OrBAC para *flows* pode-se ter em conta uma política da classe de controlo de acesso a serviços, que determina que os utilizadores pertencentes ao perfil *Network Administrator* podem aceder ao servidor HTTP da empresa. Na Tabela 16 pode-se verificar a formalização dessa mesma política no ficheiro de leitura.

Tabela 16: Política que permite o acesso de Network Administrator a servidor HTTP

Permissão				
Identificador de perfil	**Endereço IP**	**Máscara**	**Serviço**	**Ação**
2	10.0.0.4	255.255.255.255	HTTP	NORMAL

Perante esta política deve-se saber o que significa o identificador de perfil 2, que representa o perfil *Network Administrator*. Em seguida sabe-se que o endereço IP 10.0.0.4 é o endereço do servidor HTTP e tem-se uma máscara utilizada devido ao facto de esta politica ter como objeto um endereço único. O serviço HTTP em termos de implementação traduz-se no porto de valor 80. A ação NORMAL determina que um *Network Administrator* pode aceder ao recurso da rede em questão. Para que isso aconteça, em termos práticos, tem que existir uma *flow* que permita que os pacotes provenientes de um *Network Administrator* cheguem neste caso ao servidor HTTP. Essa *flow* só permite a chegada dos pacotes, se a ação associada à mesma for uma ação do tipo *Output* (*Action Type Output*), de acordo com a estruturação do OpenFlow. Logo a implementação desta política após a leitura da mesma foi realizada a partir da criação de uma *flow* com os *match fields* ou condições: endereço IP de destino 10.0.0.4 e porto TCP 80 associado.

As restantes políticas da classe de controlo de acesso a serviços foram implementadas da mesma forma, especificando em cada *flow* os portos TCP referentes ao protocolo em questão. Alguns dos protocolos utilizados e respetivos portos TCP estão representados na Tabela 17.

Tabela 17: Protocolos utilizados e respetivos portos TCP

Protocolo	Porto TCP
File Transfer Protocol (FTP)	21
Hypertext Transfer Protocol (HTTP)	80
Secure Shell (SSH)	22
Simple Mail Transfer Protocol (SMTP)	25
Telnet	23
VoIP	5060

Na classe de políticas de Qualidade de Serviço, a distribuição da largura de banda na rede empresarial foi realizada utilizando *Meters* OpenFlow, que permitem uma monitorização do *bit rate* a que o tráfego é transmitido [22]. Criaram-se *flows* que direcionam os pacotes para um meter usando a instrução OpenFlow *goto-meter*, onde o meter pode depois realizar uma operação baseada no *rate* a que recebe os pacotes. No contexto do nosso modelo, utilizaram-se os *meters* para limitar a largura de banda disponível para cada utilizador consoante o perfil em que se inserem.

Dentro da classe de políticas de Qualidade de Serviço incorporaram-se também um outro tipo de políticas: políticas que garantem uma ligação *Highspeed* para o utilizador. Isto foi implementado, sabendo que existem portas nos *switches* HP que garantem um *bit rate* superior ao das restantes portas. Quando foi definido que deveria existir uma ligação *Highspeed* entre uma máquina do perfil *Research & Development* e o servidor HTTP dedicado a *developers* instalou-se uma *flow* que fizesse com que os pacotes com origem numa máquina R&D e destino servidor, fossem enviados por uma das portas dos *switches* HP que garantem maior largura de banda. Antes de o pacote ser enviado por essa porta testa-se a disponibilidade da mesma. Ou seja, sendo as portas com maior *bit rate* nos *switches* da HP a porta 25 e a 26, testa-se se é possível chegar ao servidor HTTP para *developers* neste caso, através de uma dessas portas. Então, a existência de ligações *Highspeed* na rede parte do pressuposto de que neste caso o servidor HTTP para *developers* foi fisicamente ligado à porta 25 ou à porta 26.

A implementação das políticas de permissão ou bloqueio de acesso a Websites, ou seja, da terceira classe de políticas que se criaram, foi possível através da criação de uma *flow* que encaminha todos os pacotes para o controlador, para que estes sejam analisados. Quando um pedido DNS chega ao controlador, a aplicação verifica o ficheiro que contém as políticas de permissão ou bloqueio de acesso a domínios concretos. Na prática, verifica-se se o domínio contido no pacote DNS que chegou ao controlador se encontra no ficheiro que lista todos os domínios bloqueados.

Especificamente, verifica-se o perfil de utilizador que gerou o pedido DNS e se o domínio estaria bloqueado ou não para esse perfil em concreto. No caso de o domínio não estar bloqueado para esse perfil envia-se o pacote do controlador de volta para a rede. No caso de o domínio estar bloqueado o tráfego não é enviado do controlador de volta para a rede.

A classe de políticas de reencaminhamento de tráfego foi exemplificada através de uma política que define que quando existe tráfego que tem como destino algum dos servidores HTTP da empresa, esse tráfego deve passar primeiro por um *load balancer* para que seja realizado balanceamento de carga. Para materializar esta política na rede, foi criada uma *flow* que altera o destino dos pacotes que fizerem o *match* com os seus *match fields*, colocando como destino o endereço IP do *load balancer*. Chegando o tráfego ao *load balancer*, este faz chegar o mesmo aos servidores HTTP.

4 Resultados obtidos na aplicação SDN

Os resultados apresentados foram obtidos através da utilização do Mininet para simulação da rede representada na Figura 7. Exceção feita ao teste das políticas de Qualidade de serviço que foi realizado utilizando um *switch* físico da HP ligado ao controlador onde instalámos a nossa aplicação. Considerando então a forma como foram obtidos os resultados, estes foram distribuídos por quatro secções, cada uma referente a uma classe de políticas do nosso modelo.

4.1.1 Classe de políticas de controlo de acesso a serviços

Para demonstrar os resultados obtidos em termos de controlo de acesso a serviços da rede empresarial escolhemos apenas alguns exemplos que ilustram o que se pode obter com políticas desta classe.

Uma das políticas criadas impede os utilizadores do perfil *Guest* de aceder ao servidor FTP que adicionámos à nossa rede no Mininet. O servidor FTP utilizado foi o servidor Pure-FTPd [23] por ser seguro e por ser atualizado regularmente. Este servidor servia as nossas necessidades, que no fundo eram apenas de resposta a pedidos FTP.

Após a instalação deste servidor FTP no sistema onde correu o Mininet com a nossa rede empresarial, necessitámos de encontrar uma forma de enviar pedidos FTP para uma máquina específica, onde estivesse a correr o servidor FTP. Como foi definido na topologia de rede criada, o servidor FTP na empresa foi simulado no *host 7* (com endereço IP 10.0.0.7) da rede. Os pedidos para o servidor foram enviados utilizando o *software* curl [24] que fornece uma ferramenta de linha de comandos e uma biblioteca para transferir dados. Utilizámos este *software*, visto que, nos permite especificar qual o protocolo que vamos utilizar para transferir dados. A especificação do protocolo que vamos utilizar para transferir dados é feita a partir da definição do porto que pretendemos utilizar para a transferência de dados. Ou seja, para enviar um pedido de um utilizador do perfil *Guest* para o servidor FTP, devemos executar no terminal do *host 1*: *curl 10.0.0.7:21*.

O comando é executado desta forma assumindo que o *Guest* efetuou o login na máquina correspondente ao *host 1*. De notar que, utilizando o curl especificamos o destino do pedido (neste caso o servidor FTP em 10.0.0.7) e o porto a utilizar (neste caso o porto 21 correspondente ao protocolo FTP).

Na Figura 12 podemos verificar que o acesso ao servidor FTP foi bloqueado para um *Guest*, tal como foi definido numa das nossas políticas. Obtivemos a mensagem *Failed to connect* dado que a nossa política se traduziu numa *flow* que descarta os pacotes com origem no *host 1*, que tenham como destino o *host 7* e que estejam associados ao porto FTP. Tendo em conta que o protocolo OpenFlow não possui uma ação específica para descartar pacotes, criámos uma *flow* sem qualquer ação associada, ou seja, com uma *action list* vazia. Esta *flow* é criada aquando da autenticação de um utilizador do perfil *Guest*.

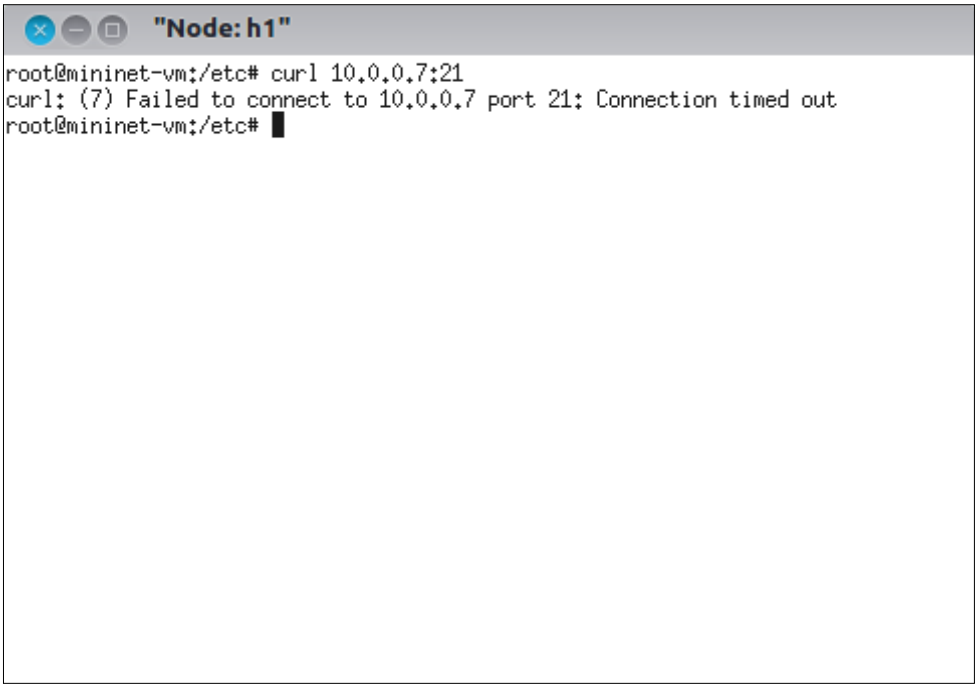

Figura 12: Bloqueado acesso de Guest ao servidor FTP

O acesso interno na rede foi testado com recurso ao protocolo ICMP, utilizando o comando *ping* para verificar se o utilizador consegue chegar à máquina de destino. Este foi um dos testes mais importantes, visto que, os utilizadores do perfil *Guest* devem ter o acesso negado a todas as máquinas da rede porque os membros desse perfil não são funcionários da empresa.

A política que define este bloqueio de acesso interno total, foi traduzida numa *flow* que é instalada nos *switches* da rede quando um *Guest* se autentica e que tem num dos seus *match fields* o protocolo ICMP. Nessa *flow* a gama de endereços e a máscara especificados englobam todas as máquinas da rede, para que um *Guest* tenha acesso bloqueado a toda a rede. Na Figura 13, temos a utilização do comando *ping* a partir de um *host 1* autenticado como *Guest* e o acesso bloqueado sendo que nenhum pacote chegou ao destino como seria de esperar.

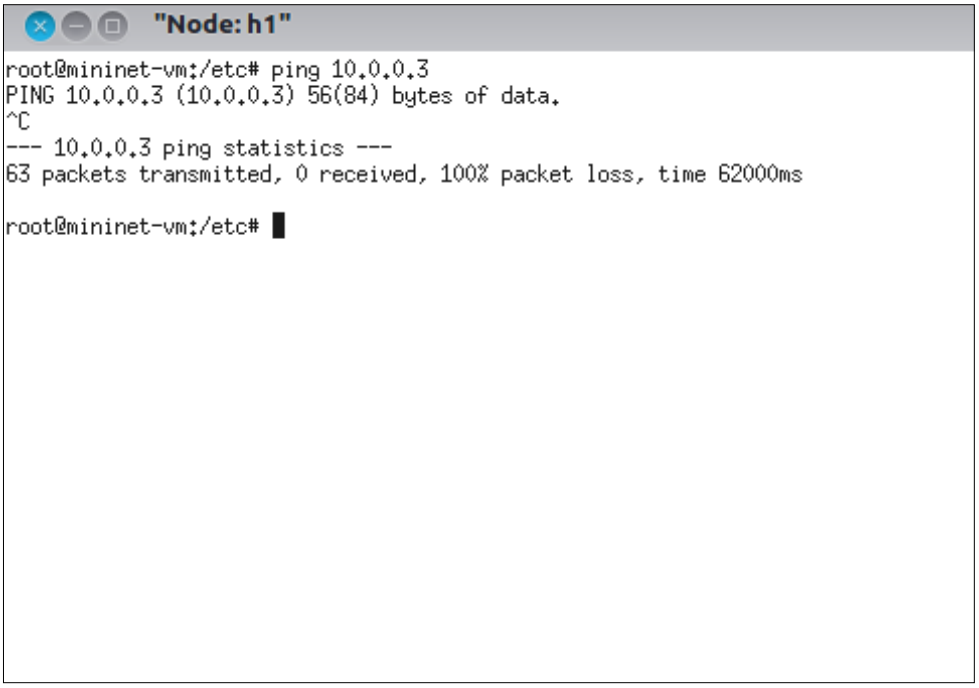

Figura 13: Bloqueado tráfego ICMP de um Guest

O acesso de um utilizador do perfil *Network Administrator* ao servidor FTP é demonstrado na Figura 14 para podermos exemplificar a permissão de um acesso, ao invés de um bloqueio como acontece para o perfil *Guest*. Podemos verificar a existência da mensagem 530 do Pure-FTPd que surge devido ao facto de o utilizador não se ter autenticado perante o servidor FTP com um nome de utilizador e uma *password* definidos na configuração do mesmo. O servidor FTP não foi configurado porque apenas nos interessava testar se o acesso ao mesmo estava ou não bloqueado, para um determinado perfil, neste caso o perfil de *Network Administrator*. Como é natural, numa rede real o servidor FTP em questão teria de ser configurado.

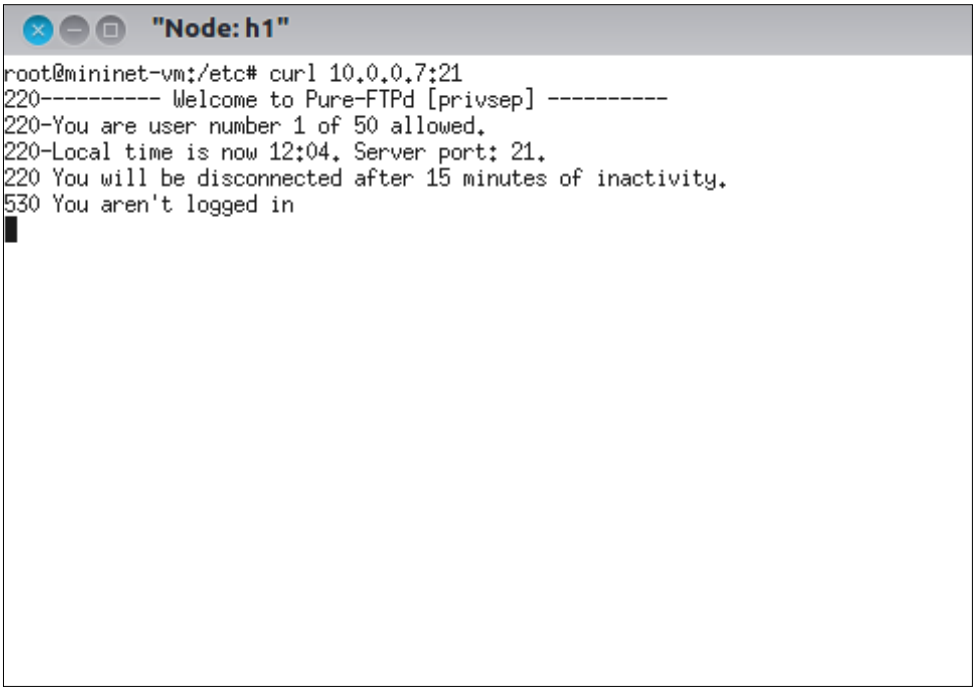

Figura 14: Permitido acesso de Network Administrator ao servidor FTP

Como foi referido, adicionámos à nossa rede um servidor HTTP que apenas pode ser acedido pelos membros do perfil *Research and Development.* A implementação de políticas como a que define que apenas um perfil pode aceder ao servidor HTTP foi desenvolvida da mesma forma que as políticas de permissão e bloqueio de tráfego FTP, através da especificação do porto utilizado pelo protocolo em questão, no conjunto de *match fields* da *flow*.

O SimpleHTTPServer [25] foi o servidor HTTP escolhido para realizar os testes, um módulo de um servidor Web desenvolvido em Python que permite o tratamento de pedidos GET e HEAD. Este foi o servidor HTTP escolhido pela sua simplicidade de utilização. Na prática só precisávamos de um servidor que respondesse a pedidos HTTP. Com a utilização do SimpleHTTPServer não foi necessário realizar configurações e bastou-nos instalar o Python na máquina virtual onde corremos o Mininet. Neste caso, como no pedido HTTP não foi especificado nenhum ficheiro HTML em concreto o SimpleHTTPServer chamou o método *list_directory()* para gerar uma *directory listing*, que foi recebida no *host 1* como se pode verificar na Figura 15.

Figura 15: Permitido o acesso de R&D ao Servidor HTTP

O acesso a uma base de dados destinada apenas aos utilizadores do perfil *Accounting* (que corresponde ao *host 6* da nossa topologia) também foi testado utilizando o comando *ping.* Na Figura 16 temos o exemplo de uma tentativa de acesso a essa base de dados feita por um *Employee*. De acordo com a política criada para o seu perfil, podemos verificar que o perfil *Employee* não tem acesso a esta base de dados.

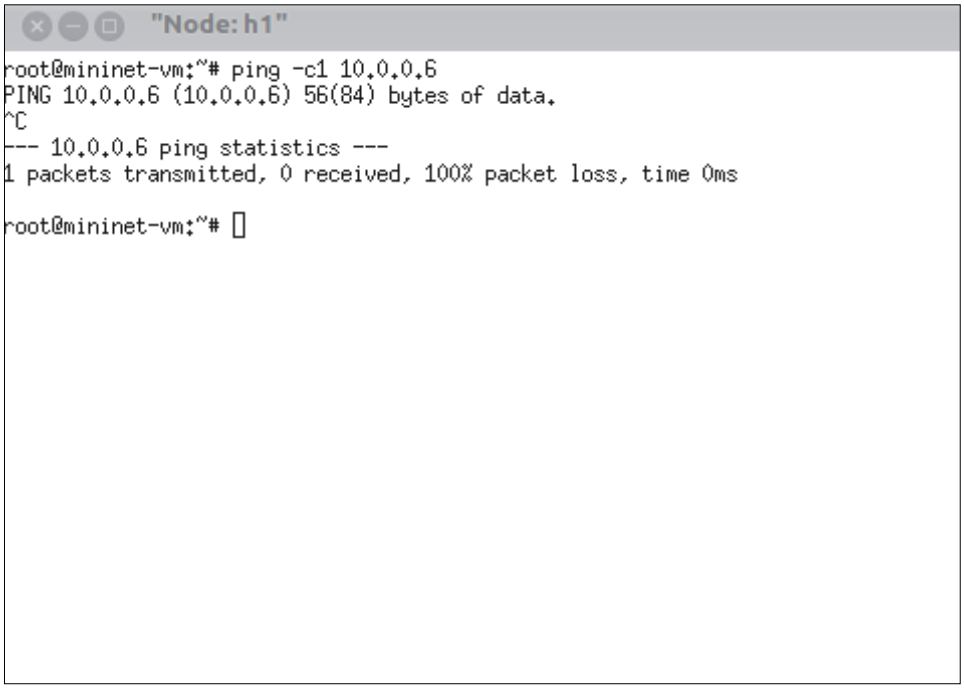

Figura 16: Bloqueado acesso do perfil Employee à base de dados de Accounting

Na Figura 17 temos o exemplo do único caso em que o acesso à base de dados é permitido, o que acontece quando o pedido provém de um utilizador autenticado e que tem o perfil *Accounting* a si atribuído.

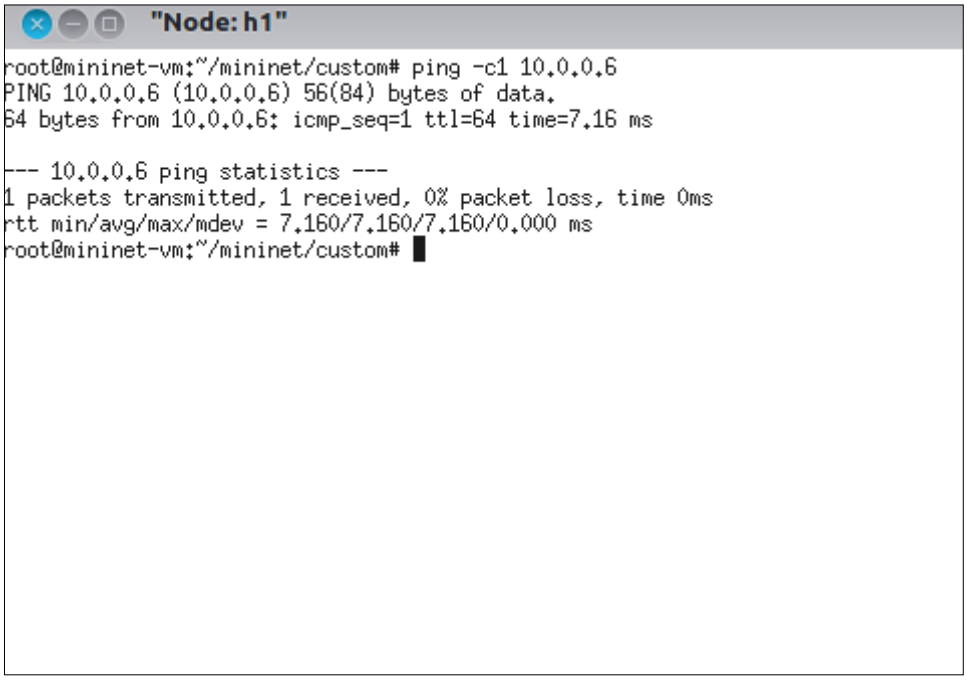

Figura 17: Permitido acesso do perfil Accounting à base de dados Accounting

4.1.2 Classe de políticas de Qualidade de Serviço

Na realização de testes para determinar a largura de banda para cada utilizador consoante o seu perfil, foi realizada uma simplificação. Como já foi referido, assumimos a título de exemplo uma largura de banda total para distribuir por todos os membros da empresa de 1 Gigabit por segundo. A partir daí efetuamos uma distribuição dessa mesma largura de banda de acordo com as necessidades dos membros de cada perfil. Ao efetuar a distribuição da largura de banda, quando na rede disponibilizamos uma largura de banda total de 100 Mbps para o conjunto dos membros do perfil *Accounting*. Devemos ter em conta que a largura de banda atribuída a cada utilizador do perfil *Accounting* depende do número total de membros desse perfil. Logo, para efeitos de simplificação durante os testes, assumimos que existe na empresa apenas uma máquina por departamento ou perfil. Dessa forma, quando um utilizador se autentica como pertencendo ao perfil *Accounting* a largura de banda disponível será de 100 Mbps. A mesma simplificação é válida para os restantes perfis.

Estes valores de largura de banda foram especificados diretamente utilizando *Meters*. Na prática todas as *flows* criadas tiveram um *Meter* associado e quando cada *Meter* foi criado, foi-lhe atribuído um número de identificação de acordo com a ordem pela qual considerámos os perfis ao longo do desenvolvimento do modelo de controlo de acesso, desde a fase de criação de perfis e políticas.

Foram efetuados testes para determinar se os utilizadores autenticados na rede tinham uma largura de banda atribuída de acordo com as políticas de especificação de qualidade de serviço que foram

criadas. Testes esses feitos utilizando a ferramenta Iperf. Iperf [26] é uma ferramenta que possibilita medir ativamente a largura de banda máxima que é possível atingir em redes IP, podendo utilizar os protocolos TCP e UDP. Para testar a largura de banda que um determinado utilizador pode ter na nossa rede, utilizámos um dos *switches* HP, com um AP ligado a este mesmo *switch*- A conexão do utilizador foi realizada através do AP. Para além do dispositivo através do qual o utilizador se autenticou, ligámos também um computador a uma das portas do *switch*. Este computador teve como finalidade correr um servidor Iperf, para podemos estabelecer uma ligação através da ferramenta Iperf, do utilizador autenticado ao computador onde corria o servidor Iperf. Na nossa topologia já tínhamos um computador que desempenhava a função de servidor RADIUS, por isso utilizámos essa máquina também para correr o servidor Iperf. Neste caso tivemos então a máquina que se ligou através do *Access Point* a desempenhar o papel de cliente Iperf e a máquina ligada diretamente a uma porta do *switch*, o servidor RADIUS, a desempenhar o papel de servidor Iperf. De notar que tipicamente, ao utilizar o Iperf, o cliente liga-se ao servidor a partir do porto 5001 e que a largura de banda mostrada pelo Iperf corresponde à ligação estabelecida do cliente para o servidor.

O comando Iperf tem vários argumentos que dão a possibilidade ao utilizador de retirar vários tipos de informação e de efetuar vários tipos de testes. Podemos executar o comando Iperf definindo argumentos para fazer a medição da largura de banda bidirecional, para definir o tamanho da janela TCP, para especificar o porto através do qual a ligação é estabelecida, para utilizar o protocolo UDP em vez do protocolo TCP entre outras possibilidades.

Devido ao facto de estarmos apenas preocupados em garantir que o valor de largura de banda da ligação entre o utilizador e o servidor estava dentro dos valores espectáveis para o seu perfil, não utilizámos nenhum argumento adicional ao executar o comando Iperf. Especificámos apenas os argumentos estritamente necessários, no nosso caso o argumento –c ou –s conforme o objetivo seja criar um cliente ou um servidor Iperf respetivamente. Na utilização do comando com o argumento –c foi especificado também o endereço IP do servidor para que a ligação possa ser estabelecida com sucesso.

Na Figura 18, temos uma representação simplificada da topologia utilizada para realizar os testes, tal como foi descrita nos parágrafos anteriores.

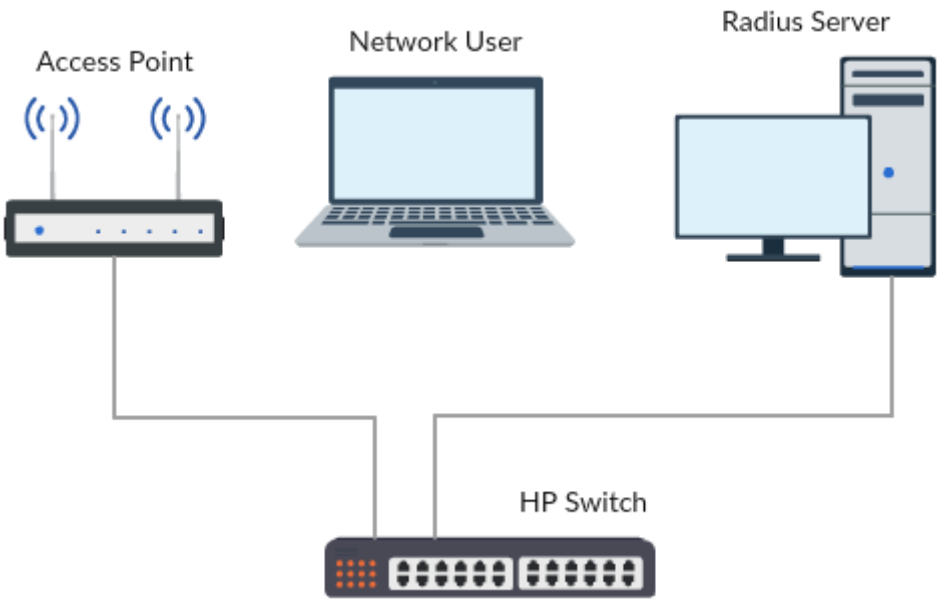

Figura 18: Topologia para testes de largura de banda

Esta topologia permitiu-nos realizar os testes de largura de banda, em que os valores obtidos para cada perfil estiveram sempre abaixo dos valores definidos através dos *Meters*.

Nas Figuras 19, 20 e 21 estão representados alguns dos resultados obtidos. Optámos por não listar todos os resultados obtidos, mas apenas alguns que permitem verificar a largura de banda de dois perfis em específico.

Na Figura 19 podemos verificar o resultado de um teste, realizado utilizando a ferramenta Iperf, para um utilizador autenticado como pertencente ao perfil *Guest*. Como foi referido, os membros deste perfil têm uma largura de banda limitada a 50 Mbps.

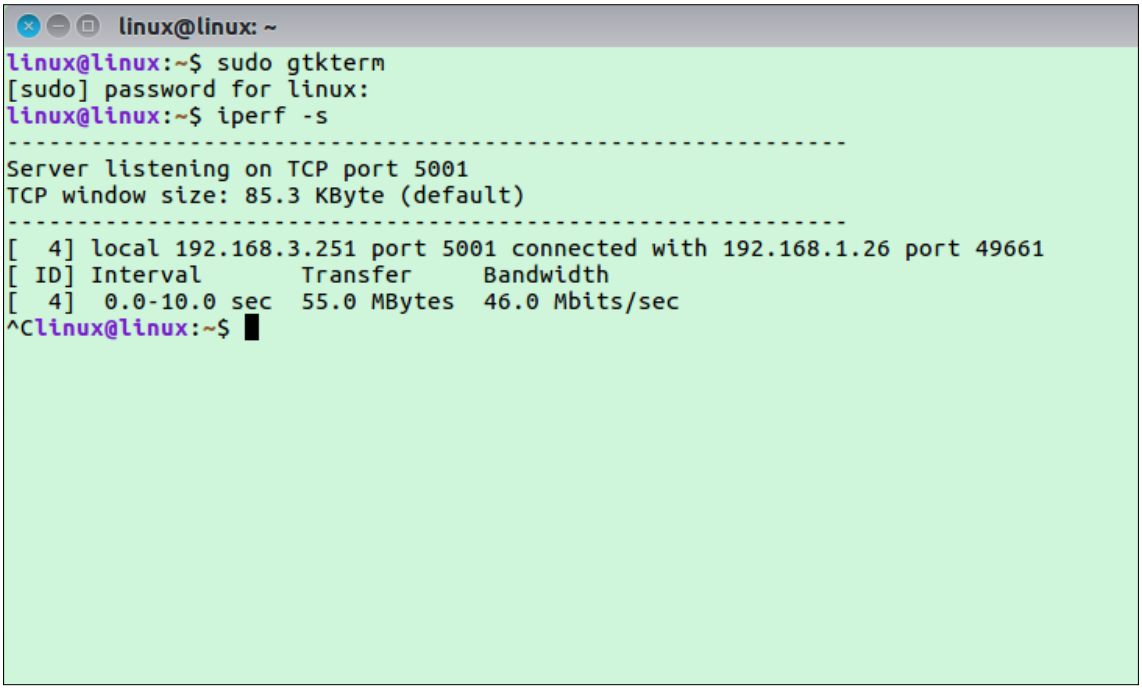

Figura 19: Largura de banda de perfil Guest

Podemos assim, a partir da Figura 19 e da Figura 20 verificar o correto funcionamento dos *Meters* criados para fazer uma gestão da largura de banda disponível, gestão essa baseada em perfis. Na Figura 20 temos um segundo teste que foi realizado para o mesmo utilizador do perfil *Guest* para confirmar que existe consistência nos valores de largura de banda apresentados. Para além de os valores que obtivemos nos dois testes serem coerentes com a largura de banda disponível para um Guest, de 50 Mbps, verificamos também que existe uma variação mínima entre os valores obtidos nestes dois testes.

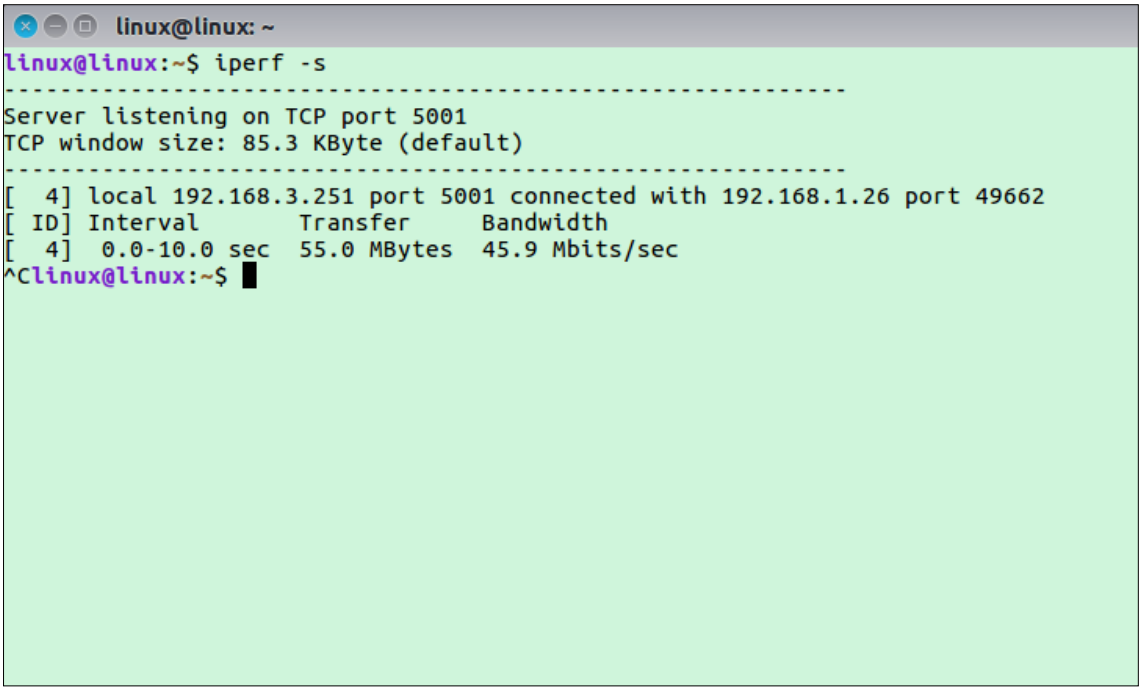

Figura 20: Segundo teste de largura de banda de perfil Guest

Sendo o perfil *Employee* para utilizadores que pertencem à empresa, ao contrário dos utilizadores *Guest*, a largura de banda atribuída a qualquer *Employee* é de 100 Mbps. Isto pode ser verificado pelo teste da Figura 21 em que foi obtida uma largura de banda de 83,9 Mbps, abaixo dos 100 Mbps que limitam a largura de banda disponível para este perfil.

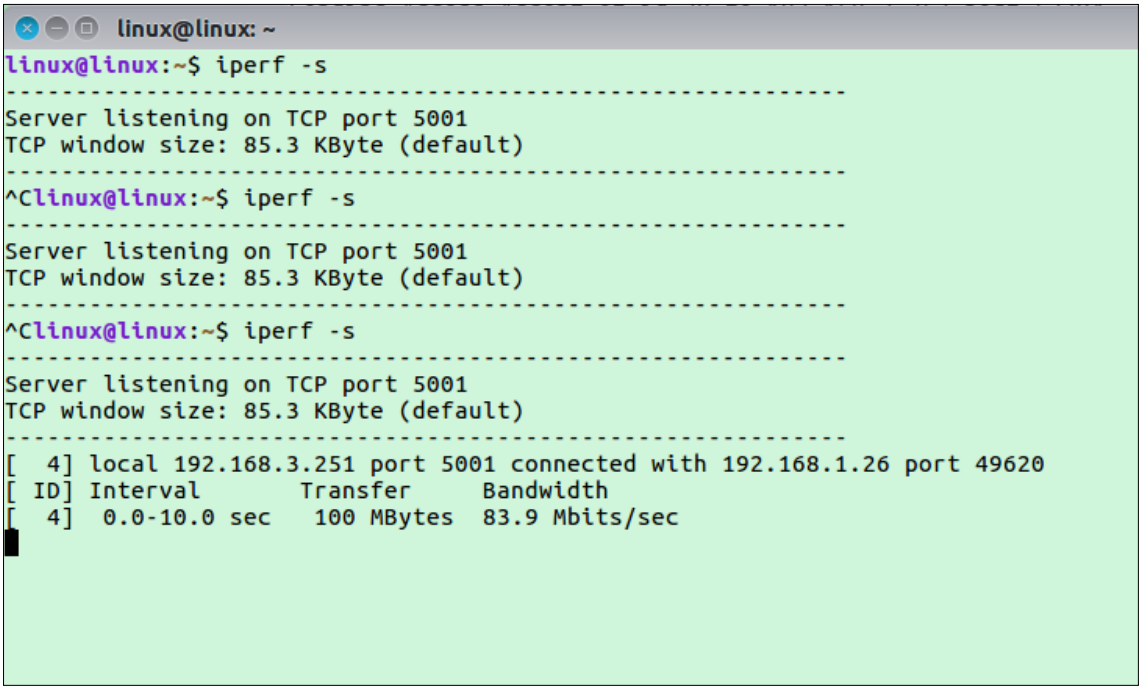

Figura 21: Largura de banda de perfil Employee

Os restantes perfis foram testados com a mesma topologia e foram obtidos valores concordantes com os que foram estabelecidos, para cada perfil, através da utilização de *meters*.

4.1.3 Classe de políticas de bloqueio de acesso a Websites

A implementação das políticas de bloqueio de acesso a Websites foi desenvolvida com o auxílio de um ficheiro onde listámos os domínios bloqueados para cada perfil. No fundo traduzimos este tipo de políticas para um ficheiro. Instalámos nos *switches* uma *flow* que encaminha os pedidos DNS para o controlador, onde foram analisados. Verificámos se os domínios existentes no pacote DNS estavam listados no ficheiro. Se esses domínios estavam listados no ficheiro então estariam bloqueados para o perfil do utilizador que gerou o pedido DNS.

A *flow* criada para fazer os pedidos DNS chegar ao controlador é baseada no mesmo conceito das *flows* que instalámos para implementar a classe de políticas de controlo de acesso a serviços. Utilizámos a funcionalidade que o OpenFlow nos disponibiliza de criar *flows* para enviar certos tipos de tráfego para o controlador. Essa funcionalidade existe na forma de uma ação que se pode associar a uma *flow* e que especifica o envio de pacotes para o controlador. Sendo o porto UDP 53 utilizado pelo DNS, criámos uma *flow* com esse *match field* para que todo o tráfego referente a pedidos e respostas DNS fosse enviado para o controlador.

As políticas da classe de políticas de bloqueio de acesso a *websites* estiveram sujeitas a condições temporais, baseadas no conceito de *Temporal Context* do modelo OrBAC como foi referido no capítulo 3.1.3. Na prática, a condição temporal ditava que os domínios bloqueados não seriam bloqueados fora do horário de trabalho dos funcionários. Ou seja, não seriam bloqueados durante a hora de almoço e após o fim do horário laboral. A implementação desta condição foi desenvolvida de forma simples, fazendo a verificação do domínio em questão, apenas durante o horário que definimos. A classe Java *Date* foi usada para sabermos quando um pedido DNS chega ao controlador, qual a hora atual. O funcionamento é simples, quando chega um pedido DNS ao controlador e a hora atual está fora do horário laboral, então o controlador encaminha o pacote sem executar o nosso processo de verificação do ficheiro com os domínios DNS bloqueados. Quando a hora atual se insere no período referente ao horário de trabalho, o ficheiro é verificado, tendo em conta o perfil do utilizador e o pacote só é encaminhado caso ao domínio não esteja bloqueado.

Quando partimos para a fase de testes de acesso a domínios permitidos ou bloqueados, testámos todos os domínios que entendemos bloquear para cada perfil. No entanto não apresentamos aqui o resultado de todos esses testes e decidimos mostrar apenas alguns. Os suficientes para que se verifique o funcionamento do mecanismo de controlo de acesso a domínios da internet.

Podemos verificar na Figura 19 que o acesso ao domínio www.facebook.com foi bloqueado com sucesso, para utilizadores do perfil *Employee*. Para testar o acesso a domínios específicos utilizámos a ferramenta nslookup [27] que permite obter informação sobre os registos do domínio que especificarmos, no fundo permite descobrir a que endereço IP corresponde um determinado *host name*.

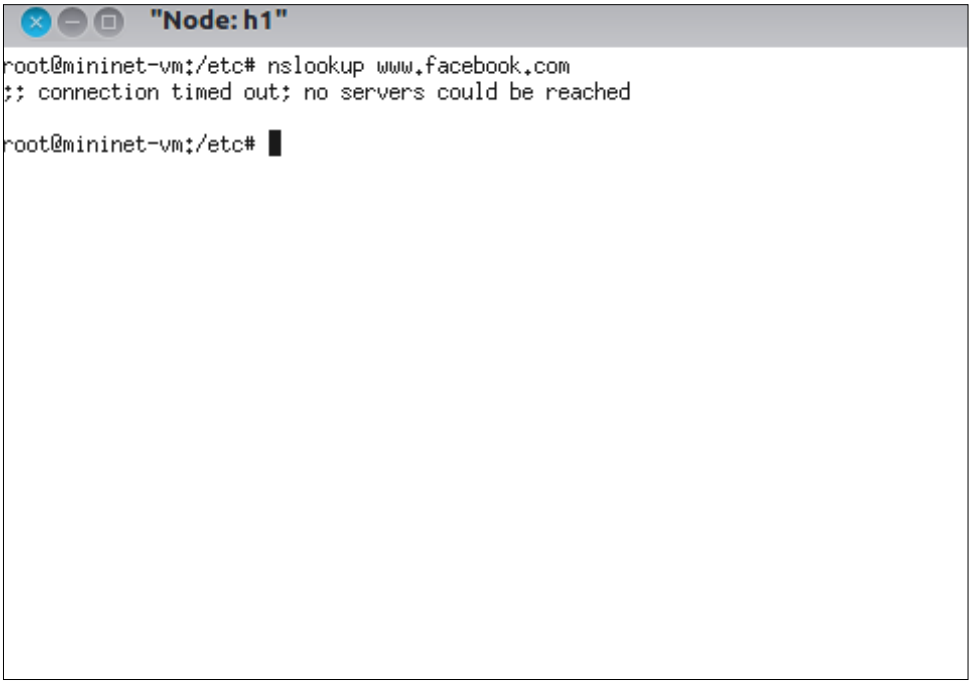

Figura 22: Bloqueado acesso do perfil Employee ao domínio www.facebook.com

Na Figura 20 temos o exemplo de informação obtida sobre o domínio www.google.com. Informação que conseguimos obter devido ao facto de este não ser um dos domínios bloqueados pelas nossas políticas e que foi obtida através do comando nslookup num terminal do *host 1* previamente autenticado como utilizador do perfil *Employee*.

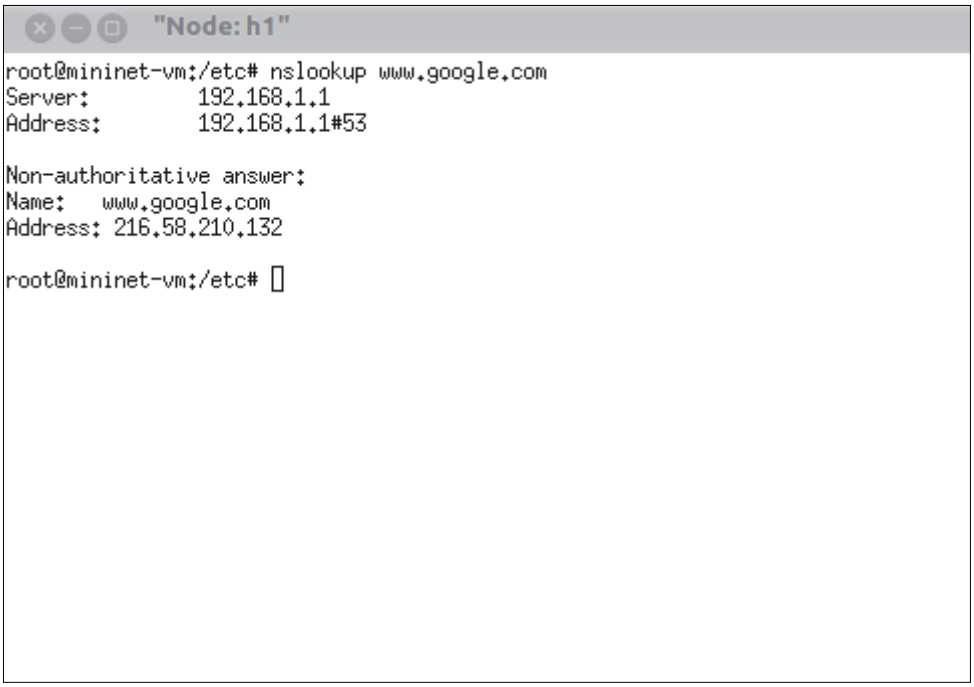

Figura 23: Permitido acesso do perfil Employee ao domínio www.google.com

4.1.4 Classe de políticas de reencaminhamento de tráfego

Na implementação de políticas de reencaminhamento de tráfego foi necessário fazer com que algum tipo de tráfego fosse encaminhado para uma máquina específica. Para atingirmos esse objetivo, criámos *flows* com ações de envio de pacotes para um endereço MAC. Ou seja, quando implementámos, por exemplo, uma política que encaminha todo o tráfego HTTP para um *load balancer*, admitimos que um endereço MAC seria o do *load balancer* que estaria a funcionar uma rede real e criamos uma ação para enviar o tráfego para esse endereço MAC. Na nossa rede de testes fixámos o endereço do *load balancer* como sendo o endereço 00:00:00:00:00:13. Como não tivemos possibilidade de trabalhar com um *load balancer* real, simplificámos o processo de testes assumindo que um dos *hosts* da nossa topologia de rede empresarial, criada no Mininet, representaria para o nosso ambiente de desenvolvimento e de teste, o que na prática seria um *load balancer*. Assumimos que fazendo chegar os pacotes ao *load balancer*, este faria chegar os mesmos aos servidores HTTP, fazendo uma distribuição correta do tráfego pelos servidores. Partindo destas premissas bastou-nos verificar se o tráfego HTTP chegava ao *host 13* da rede Mininet, que representou o *load balancer*, através da utilização da ferramenta de análise de pacotes tcpdump [28]. Como podemos constatar na Figura 21 chegou ao *host 13* tráfego com origem na máquina com endereço IP 10.0.0.1 e destino 10.0.0.4. Isto porque para realizar este teste um utilizador do perfil *Employee* autenticou-se no *host 1* e enviou um pedido para o servidor HTTP que estava a correr no *host 4*.

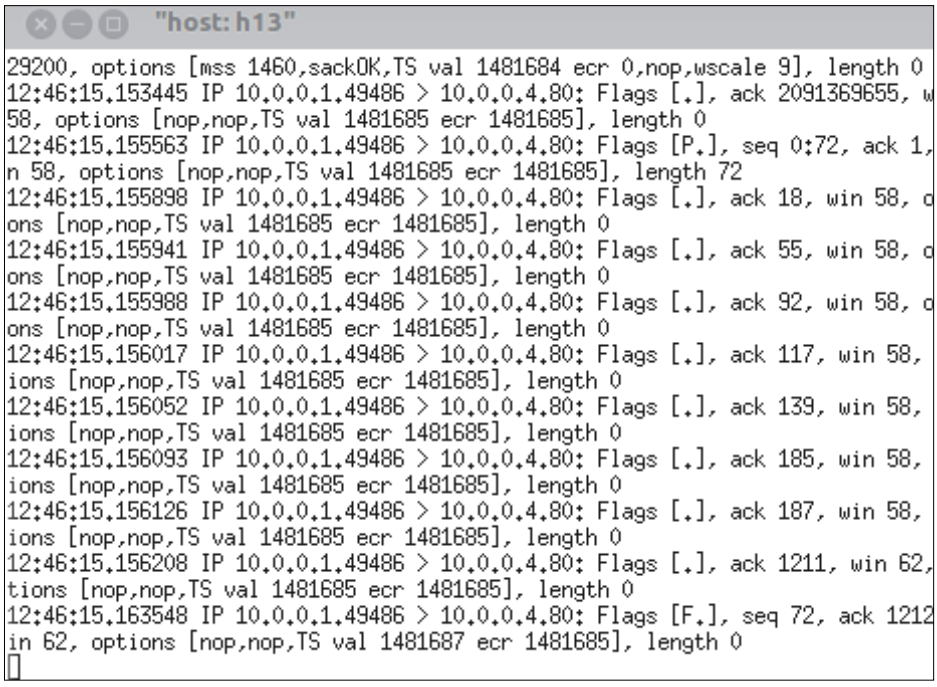

Figura 24: Pedido HTTP de um Employee a passar no load balancer

5 Conclusões e Trabalho Futuro

Nos dias que correm, o *Software Defined Networking* é cada vez mais uma área relevante, no que diz respeito à configuração e gestão de redes. Num momento em que está a ser realizada a transição das redes de computadores tradicionais para as *Software Defined Networks*, o controlo de acesso é, tal como sempre foi, uma questão fulcral para o desenvolvimento de bons sistemas de gestão de redes. A flexibilidade introduzida pelas tecnologias SDN permite alterar o paradigma do controlo de acesso propriamente dito, na medida em que este pode ser realizado de uma forma mais dinâmica através da alteração e criação de políticas que podem ser feitas em qualquer momento. A relevância deste projeto tornou-se então evidente, visto que não pode existir uma boa solução de configuração e gestão de redes que não realize controlo de acesso e, para além disso, o SDN criou novas oportunidades em relação à forma como o controlo de acesso pode ser realizado.

Este projeto foi pensado de forma a englobar as várias fases de desenvolvimento de uma solução de controlo de acesso e, partindo da criação de um modelo de controlo de acesso teórico, acabou por culminar na implementação prática das políticas presentes nesse modelo. Implementação essa que foi realizada utilizando *software* e *hardware* próprios para o domínio do SDN, desenvolvidos e comercializados pela HP e que permitem a implementação de uma solução deste tipo de forma simples e rápida.

Para além do facto de este projeto permitir percorrer todo este caminho de desenvolvimento de uma solução SDN, também foram incorporados mecanismos e políticas de gestão de redes que ultrapassam a abrangência do controlo de acesso, fazendo com que este projeto se torne relevante para quem procura conhecimento fora do âmbito do controlo de acesso mas dentro da área do SDN.

Transversalmente a todos os tipos de políticas criadas temos a importância de estas terem sido testadas em laboratório não ficando apenas no campo da teoria e das possibilidades. Muitas dessas políticas foram assentes em conceitos básicos, mas isso não retira relevância ao projeto dado que estamos ainda numa fase de introdução do SDN num contexto mais massificado e nesse sentido as primeiras soluções de controlo de acesso a serem implementadas devem ser práticas e simples, para que depois possamos partir para outras de maior complexidade.

Um dos objetivos do projeto foi também o de criar mais um trabalho que abra a discussão das vantagens e desvantagens do SDN na atualidade. Foi possível criar um projeto que pode servir para quem tenha como objetivo a implementação de uma solução deste tipo numa rede real. Bastando para isso a criação e definição de perfis e políticas baseados nos que foram estruturados neste projeto. Sendo que os que apresentámos neste projeto foram pensados de forma a transmitirem ideias gerais do que pode ser interessante, não especificando demasiado para um tipo de redes concreto.

Nesse sentido, espera-se que este projeto possa servir como base para a implementação de algumas soluções deste tipo e para a cimentação do *Software Defined Networking* no mundo académico e no mundo profissional. Aumentando a complexidade do sistema, criando mais perfis, mais políticas e também introduzindo mais conceitos que não se enquadrem no controlo de acesso mas que possam ser relevantes em contextos muito específicos, dependentes da rede que se pretende gerir e configurar.

Referências

[1] A. Hakiri, A. Gokhale, P. Berthou, D. C. Schmidt, and T. Gayraud, "Software-defined networking: Challenges and research opportunities for future internet," *Comput. Networks*, vol. 75, no. PartA, pp. 453–471, 2014.

[2] D. Kreutz, F. M. V Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[3] Open Networking Foundation, "Principles and Practices for Securing Software - Defined Networks," no. January, 2015.

[4] F. Hu, Q. Hao, and K. Bao, "A survey on software-defined network and OpenFlow: From concept to implementation," *IEEE Commun. Surv. Tutorials*, vol. 16, no. 4, pp. 2181–2206, 2014.

[5] N. Networks, C. Nexus, and C. Xenserver, "Open vSwitch" [Online]. Available: http://openvswitch.org/. [Accessed: 16-Feb-2016].

[6] "POForwarding." [Online]. Available: http://www.poforwarding.org/. [Accessed: 16-Feb-2016].

[7] A. Opflex, "OpFlex : An Open Source Approach." [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/data-center-virtualization/application-centric-infrastructure/white-paper-c11-731304.html. [Accessed: 16-Feb-2016].

[8] M. Jammal, T. Singh, A. Shami, R. Asal, and Y. Li, "Software defined networking: State of the art and research challenges," *Comput. Networks*, vol. 72, pp. 74–98, 2014.

[9] D. Kreutz, F. M. V. Ramos, and P. Verissimo, "Towards secure and dependable software-defined networks," *Proc. Second ACM SIGCOMM Work. Hot Top. Softw. Defin. Netw. - HotSDN '13*, p. 55, 2013.

[10] A. Nayak and A. Reimers, "Resonance: dynamic access control for enterprise networks," *Wren*, pp. 11–18, 2009.

[11] D. Mattos, L. Ferraz, and O. Duarte, "AuthFlow: Authentication and Access Control Mechanism for Software Defined Networking," *XXXII Simpósio Bras. Redes Comput. e Sist. Distrib. - SBRC'2014*, 2014.

[12] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, "Ethane: taking control of the enterprise," *Sigcomm '07*, pp. 1–12, 2007.

[13] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, and G. Gu, "A security enforcement kernel for OpenFlow networks," *Proc. first Work. Hot Top. Softw. Defin. networks - HotSDN '12*, p. 121, 2012.

[14] "Introduction | OrBAC: Organization Based Access Control." [Online]. Available: http://orbac.org/?page_id=21.

[15] "What is MapReduce? - Definition from Techopedia," *Techopedia.com*. [Online]. Available: https://www.techopedia.com/definition/8548/network-administrator.

[16] "MotOrBAC: an OrBAC policy editor." [Online]. Available: http://motorbac.sourceforge.net/.

[17] RDF Working Group, “RDF - Semantic Web Standards,” *W3C Recommendation February 2014*, 2014. [Online]. Available: http://www.w3.org/RDF/.

[18] “RDFa.” [Online]. Available: https://rdfa.info/.

[19] F. van H. Deborah L. McGuinness, “Owl web ontology language overview,” *W3C recommendation 10.2004-03*, 2004. [Online]. Available: https://www.w3.org/TR/owl-features/.

[20] E. Prud’hommeaux and A. Seaborne, “SPARQL Query Language for RDF,” *W3C Recommendation*, 2008. [Online]. Available: http://www.w3.org/TR/rdf-sparql-query/.

[21] W3C, “W3C Semantic Web Activity,” 2011. [Online]. Available: https://www.w3.org/2001/sw/.

[22] “How to Use OpenFlow Meters - Floodlight Controller - Project Floodlight.” [Online]. Available: https://floodlight.atlassian.net/wiki/display/floodlightcontroller/How+to+Use+OpenFlow+Meters#HowtoUseOpenFlowMeters-Whataremeters?

[23] F. Denis, “Pure-FTPd.” [Online]. Available: https://www.pureftpd.org/project/pure-ftpd. [Accessed: 12-Dec-2016].

[24] “curl.” [Online]. Available: https://curl.haxx.se/. [Accessed: 29-Dec-2016].

[25] “20.19. SimpleHTTPServer — Simple HTTP request handler — Python 2.7.13 documentation.” [Online]. Available: https://docs.python.org/2/library/simplehttpserver.html. [Accessed: 13-Dec-2016].

[26] Iperf.fr, “iPerf - The TCP, UDP and SCTP network bandwidth measurement tool,” 2016. [Online]. Available: https://iperf.fr/. [Accessed: 20-Dec-2016].

[27] “KLOTH.NET - NSLOOKUP - DNS Look up - Find IP Address .” [Online]. Available: http://www.kloth.net/services/nslookup.php. [Accessed: 29-Dec-2016].

[28] tcpdump, “Tcpdump/Libpcap public repository,” 2016. [Online] Available: http://www.tcpdump.org/ [Accessed: 27-Dec-2016]